PROCESSOR TECHNOLOGY
ACCESS.
75¢ per issue
Published by Wible/Rampton Advertising, San Francisco, Volume One, Number Two, April 1977

A Letter from the Editor

"Everyone should believe in something. I believe I'll have another drink." — Steele's Plagiarism of Somebody's Philosophy

Welcome to issue #2 of ACCESS. I hope the first one was able to provide you with something useful, after waiting so patiently for delivery. Wasn't entirely our fault this time, honest; we now know from personal experience that mailing services have their gremlins too. Anyway, issue #2 is now here and filled with goodies we hope you can use on your pet project.

One thing issue #2 isn't filled with is feedback from you. No way it could be, since it went to press before most of you ever got your hands on #2. We really do want to facilitate an exchange of ideas, comments, gripes, what have you, so please get those cards and letters into us. Are you more interested in hardware stuff than software stuff? Vice versa? What bugs bug you? What do you do with your Sol anyway? Got any hot tips on interfacing with exotic equipment? Just want to sound off about something? Or maybe even say something nice about us? We definitely want issue #3 to have more of you in it, instead of just us talking to the walls.

Speaking of us, you'll be reading this just about the time of the 1st West Coast Computer Faire. Come give us some feedback in person; we'll be there along with all the competition, and we'll have some good stuff to show you. You might even get to see me if you look closely; I'm the one wearing the red suspenders. For those of you who can't make it to San Francisco, we'll have a report in ACCESS #3.

Got to sign off now and fix that Sol PCB that just came in for warranty repair. Probably sabotage — whoever heard of a SOL not working?)

Aram Attarian II

Subscription Information

ACCESS is published every six weeks. If you like what you see, we hope you'll send us $4.00 for a year's subscription so we can keep the info coming. Write to us at Processor Technology, 6200 Hollis Street, Emeryville, CA 94608.

Inside Emeryville!

One to One Communication

We're going to make this column a regular feature, under the peerless direction o f our Customer Service Manager, the world renowned Ralph I. Palsson (applause). He'll keep you informed on what's happening in the world of availability, delivery dates, and other such precious tidbits, and do his best to keep the lines of communication open. Good luck, Ralph. (A.A. II, Editor)

A Customer Service Department has only one reason for its existence: to fix the responsibility on someone for keeping the communication lines open, so the company doesn't become just another faceless entity, vaguely malevolent to your attempts to pursue the love and lore of computing. The someone's me, Ralph Palsson; I'm here to welcome your telephone and letter inquiries and provide you with immediate, personal service. Whether it be pre-sales information, placing an order, finding the nearest Processor Technology dealer, checking on the availability of existing or forthcoming products-we'll do our best to help.

Are you interested in more information about a product than you could glean from the catalog description? How about the VDM-1 kit you ordered way back when and still haven't received? Even the best of us make occasional errors, and letting us know about your problems is the first step towards their solution. Letters are now being sent regularly to advise you, our customers, of anticipated production and delivery schedules. PTC dealers are being kept informed too, so you can call them for on-going status reports. We're trying continually to expand and improve the Customer Service Department with the aim of giving even more efficient, reliable service. Some of the ways we're doing it:

Teletroubleshooting. Having a problem with a PTC kit? Is your Sol-20 displaying 0's and 9's and nothing else? Need technical advice on interfacing? One of our applications engineers is available

daily for phone consultation and technical assistance. Call (415) 652-8080 between 9:30 am and noon or 1:30 and 4:00 pm Pacific Time. Technically oriented software questions should go to our new division, Software Technology, at (415) 349-8080. (That's right, 8080!)

When you make a technical call about a malfunctioning kit, it helps if you take time to think out your questions first, maybe make a few notes. It's also a good idea to have your kit manual handy, and if possible, the recalcitrant piece of equipment. All calls are taken on a first come, first serve basis, so if you do have to hold for a few minutes, please be assured that you're not running up your bill for the benefit of someone who called in later. If you do run into a wait, you can also ask us to call you back collect. We will.

Again, let us remind you that if you purchased your PTC product through a dealer, try him/her first. They're all authorized to service the equipment they sell, precisely because they can give you more personalized, faster service than anyone can by mail or phone.

Need a defective part replaced? Again, the best way to handle this is through your dealer. If he's out of stock, or if you purchased directly from us, send us the defective part for replacement. Similarly, if you're missing a part from a kit, call your dealer or write to us, giving as complete a description as you can of what's missing.

Finally. We are always interested in improving our one-to-one communication, and we're open to your comments, criticisms, and suggestions. Please feel free to contact us if we can be of service in any way.

Error, Error, Does Not Compute

Errors do appear from time to time in all high class publications. ACCESS is no exception — our score for the first issue was three unfortunate oversights.

1. The instructions for wiring the 3P+S for a keyboard interface (page 3). At Step 5, we told you to connect pin 7 to a point on 3P+S leading to J2 pin 12. Pin 12 is a ground point, and you don't really want to do that. Pin 7 should be connected to a point on the 3P+S leading to IC 14 pin 13.

2. The article on interfacing PT 4KRA memory boards with the Motorola M6800 (page 2). It's all right as far as it goes, but we neglected to give proper credit to the contributor. Our apologies and thanks to J. W. Schook, P O. Box 185, Rocky Point, N.Y.

3. Last, but not least, those ever-present gremlins managed to alter our mailing address to 6800 Hollis St. Now, that's really adding insult to injury. Our correct address is 6200 Hollis St., Emeryville, CA 94608.

Introducing Software Technology Corporation

We've pulled a protozoan trip and split in two: Software Technology Corp. is a separate new company formed to take over software development from Processor Technology. The idea is to give you more and better software support by not scattering our energies about, randomly.

Software Technology is three people with an impressively vast reservoir of experience in operating systems, time sharing, real time systems, compilers, interpreters, simulators, business systems, and software development. Did we leave anything out? During this transitional period, they'll be maintaining, supporting and producing all the current PTC software, plus developing some of their own. Later (not too much later), they'll be coming out with lots of good stuff designed to get the most out of Sol.

You should have your first chance to see the caliber of Software Technology developments at the Computer Faire in San Francisco. But we're not going to tell you what, because one fundamental policy of *Software Technology* is *not* to announce anything that isn't ready. Nobody likes a tease.

Meanwhile, Software is in business and even has a phone. If you have any problems with Processor Technology software you're using now, you can call their 24-hour phone line. When nobody's there, there's an answering machine to take your message, and they *will* get back to you promptly. The

number is (415) 349-8080. You're also welcome to write your questions: PO. Box 5260, San Mateo, CA 94402.

For the time being, orders, questions about delivery and of course hardware questions should still be directed to Processor Technology. (See One to One Communication)

Read on for your first example of Software Technology software.

Announcing the SOL USERS GROUP!

The Sol Users Group was recently organized by members of the Homebrew Computer Club in Palo Alto; SUG is not affiliated in any way with Processor Tech-Technology. The purposes of SUG are to exchange software and other applications, and to create standards. If plenty of interest is shown, a Sol Newsletter will be published and sent to members.

If you own or have ordered a Sol, send your name, address, phone number and ideas to:

Bill Burns
4190 Maybell Way
Palo Alto, CA 94306
(Please don't call.)

Attention, CONSOL Users

All Processor Technology software has recently been modified in a very important way that encourages standardization. If you've been looking forward to having some Sol software for your CONSOL Proms, you'll have to let us reprogram for you. There's *no charge*. Just mail your CONSOL Proms, suitably packaged, to Processor Technology; we'll take care of the rest. Turn-around time is about 2 weeks, dependent on the vagaries of the U. S. Postal Service, natch.

If you're planning to upgrade to SOLOS anyway, don't worry about it.

Editor: Aram Attarian II Publisher: Wible/Rampton Advertising, 727 15th Avenue, San Francisco, CA 94118
ACCESS is published approximately every six weeks. Subscription rate: $4 per year, from Processor Technology Corp., 6200 Hollis St., Emeryville, CA 94608.

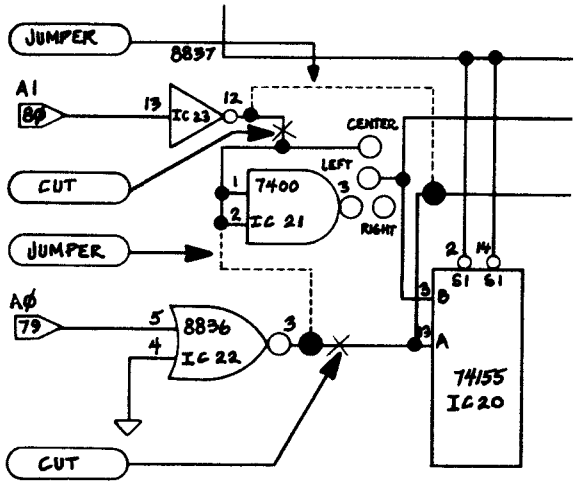
Clarkes third law: Any sufficiently advanced technology is indistinguishable from magic.

Double Your Pleasure, Double Your Fun, Or, How to Use Both Parallel Ports on the 3P+S

You can take maximum advantage of the 3P+S's versatility by making a simple modification that gives you simultaneous access to both parallel ports. Then you'll be able to use one port for a keyboard input, the other for paper tape input/output. The serial port is available for an RS232 or TTY, with status and control available at Port 0 for Processor Technology software compatibility.

The change procedure is as follows:

1. Cut the existing trace from IC 22 pin 3 to IC 20 pin 13; at IC 22 pin 3
2. Cut the existing trace from IC 23 pin 12 to IC 21 pins 1 and 2; at IC 23 pin 12
3. Run a jumper from IC 23 pin 12 to IC 20 pin 13
4. Run a jumper from IC 22 pin 3 to IC 21 pins 1 and 2
5. Select area "B" option; jumper from left to right



Assuming that area "A" board address options "00" have been selected, the ports will now be set up as follows:

Port 0 = Channel C. Control and status

Port 1 = Channel A.* Parallel data (keyboard)

Port 2 = Channel D. UART (RS232 or TTY)

Port 3 = Channel B.* Parallel data (paper tape reader)

*The strobe latches for the parallel ports remain with their respective channels.

Changing the data available status. The 3P+S interface is laid out to respond to a negative strobe input pulse low active status (i.e., FA, FB). For compatibility with Processor software, you'll want to select a high active data available status. Run a jumper from the !Q output of the respective strobe latch (i.e., AKA, AKB) to the correct status bit input in area "G." In the above configuration, the amounts to jumpering keyboard data available IC 15 pin 7 to area "G" point C6.

CUTE, CUTEST, CUTER

Below is the complete source listing for the control and monitor programs for the CUTS board. This program, CUTER, was made available with the CUTS module in the form of a cassette selling for $11.

The CUTER cassette contains object code along with a relocating loader for loading the program in any 256-byte boundary of available RAM. A new version of BASIC-5 for use with CUTER and SOLOS is also on the cassette, followed by the complete source code of CUTER. Not a bad deal for $11.

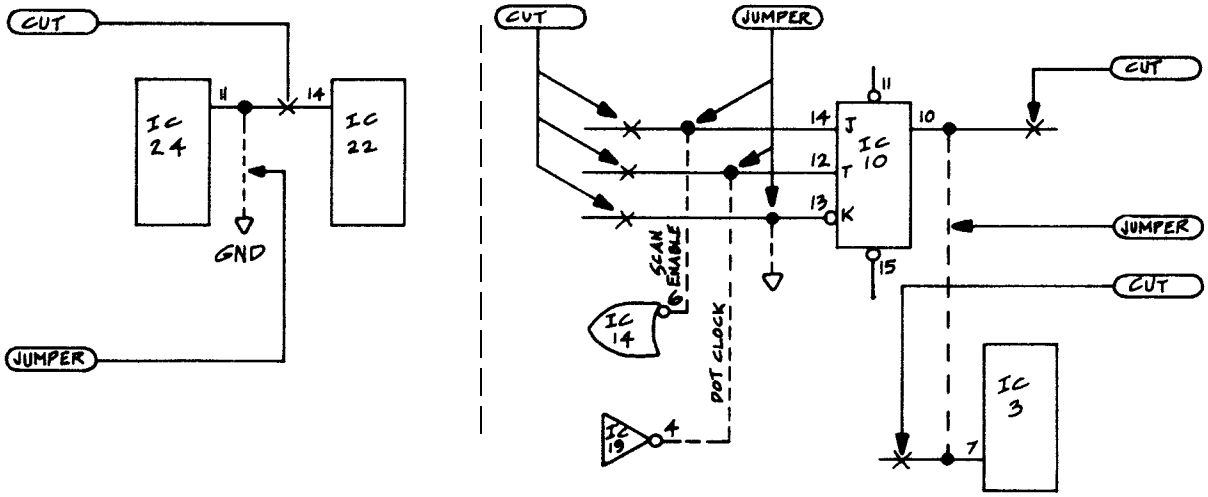
So why buy the cassette now that we're giving you the listing? It's not that we're mercenary, but consider: if you want to use PTC's software packages on cassettes; then CUTER is going to be a tremendous asset in loading those tapes, right? Right-otherwise you could spend 3 or 4 hours toggling it in from the front panel of your A#$%&R or I(*@ I.

And now, the VDM-½! Or, Modifying Your VDM-1 for 32-character Display

We're pleased to announce that unceasing technical progress has now made possible a reduction in the number of characters per line of the VDM-1 display from 64 to 32! Seriously, the modification is quite handy if you want to work with large type display, or feed RF modulated signals to a TV antenna input.

The hardware modifications are shown in the schematic drawing below. One word of caution: since we're tying the low-order address bit to ground, only character locations with bit 0 equal to 0 (i.e., even-numbered addresses), will be displayed. So if you use the modified VDM with unmodified software, you'll end up with only alternate characters on the screen and a rather cryptic message!

In the next issue of ACCESS, we'll have some spiffy software routines to use with the "VDM-½." Meanwhile, you can probably come up with some of your own. (Drop us a line if you run across anything interesting.) Just remember that a left shift of a 64-character address will give you the 32-character address, providing that you shift in a 0 to bit 0. Good luck!



1. Cut traces at:

A—pin 10 IC 10
B—pin 12 IC 10
C—pin 13 IC 10
D—pin 14 IC 10
E—pin 7 IX 3
F—pin 14 IC 22

2. Add jumpers between:

A—IC 24 pin 11 and Ground
B—IC 10 pin 10 and IC 3 pin 7
C—IC 10 pin 12 and IC 19 pin 4
D—IC 10 pin 13 and Ground
E—IC 10 pin 14 and IC 14 pin 6

FLASH - VDM Access Flicker Eliminated, Part 2

In the last issue, we gave you a hardware modification to take care of the flicker produced whenever the VDM memory is being accessed. We also promised to explore the implications a bit further this time. So here's a subroutine for the VDM driver program that implements the modification via software.

One thing to remember: if reads *from* the screen memory are needed by the driver, they have to be synchronized by a subroutine similar to this. This one only takes care of flickerless access *to* the memory.

Next issue, we'll continue the flicker saga by providing a version of the Processor PATTERN program which contains this routine for flickerless display.

```
1000 * THIS ROUTINE MOVES A CHAR. IN REGISTER B
1005 * TO VDM DISPLAY MEMORY
1010 *
1015 ***** ASSUMPTIONS:
1020 *
1025 * 1.   THE CALLER HAS SET UP A STACK
1030 *
1035 * 2.   REGISTER PAIR H&L CONTAIN THE
1040 *      VDM DISPLAY MEMORY ADDRESS
1045 *
1050 * 3.   THE ROUTINE IS RUNNING IN MEMORY
1055 *      WITH NO (0) WAIT STATES.
1060 *
1065 *
1070 *
1075 *
1080 START  PUSH   B
1085        MVI    C,2 SYNC BIT MASK (BIT 1)
1090 *
1095 *  CATCH FALLING EDGE OF SYNC
1100 *
1115 HILP   IN     0C8H   VDM STATUS PORT
1110        ANA    C      ONLY BIT 1 REMAINS
1115        JZ     HILP   UNTIL SYNC IS HI
1120 *
1125 *   SYNC IS HI. NOW WAIT
1130 *   TILL IT IS LOW
1135 *
1140 LOLP   IN     0C8H   VDM STATUS PORT
1145        ANA    C      ONLY BIT 1 REMAINS
1150        JNZ    LOLP
1155 *
1160 *  DISPLAY SWEEP IS NOW AT LEFT MARGIN
1165 *
1170        CALL   WAIT   SO SWEEP CAN MOVE TO
1175 *                     RIGHT MARGIN
1180 *
1185 *  NOW THAT SWEEP IS OFF THE SCREEN:
1190 *
1195        MOV    M,B    ACCESS DISPLAY MEMORY
1200 *
1205 *
1210        POP    B      GIVE IT BACK
1215        RET
1220 *
1225 *  THIS ROUTINE DELAYS FOR JUST ENOUGH TIME
1230 *  TO ALLOW THE SWEEP TO CROSS THE SCREEN
1235 *  *** ALTER IT WITH CARE ***
1240 *
1245 WAIT   NOP
1250        PUSH   H
1255        POP    H
1260        PUSH   H
1265        POP    H
1270        RET
```

Cann's Axiom: When all else fails, read the instructions.

This issue's Bug Squad focuses on the Sol PC board Revision D, the one you have. All the fixes described are aimed at worst case conditions, so hopefully you've never encountered the problems. But we do recommend making the changes now to forestall future headaches that might crop up if you use demanding peripherals such as discs. All the modifications described will be incorporated in the forthcoming Revision E board, so in the future we'll be designing on the assumption that all boards in the field have these fixes.

Now, you may wonder where we get the gall to blurt out that our product is not the ultimate in all respects. Quite simply, we have too much experience in product design to let ourselves get away with the attitude that goes, "We're perfect. Something must be wrong with you:" So things can be better, and here's how.

1. Clock Width Fix

The bug: Currently the width of the phase 1 (ϕ1) clock pulse is 70 nanoseconds. If you want to bring it into spec with existing 8080 chips, you should increase it to 140 nanoseconds. (8080A or 9080A are OK at 70 nsec.)

The squasher: On the top (component) side of the board, cut the trace between jumpers D and E of (U90 and U91) of the clock generator. On the bottom (solder) side of the board, connect the jumper from pin E to the feedthrough which leads to pin 5 of U91.

2. Phantom Glitch Fix

The bug: Occasionally a Sol will power up with three "phantom" cycles instead of the necessary four, causing a "crash: These are the cycles which use the "four phase wonder" software in the monitor.

The squasher: Connect a jumper on the solder side of the board as shown in Figure B. It goes from pin 4 of U76 to the feedthrough immediately below pin 1 of U76.

3. Ground Noise Fix

The bug: The paths from the bus drivers to the bus ground are too long, producing occasional ground noise.

The Squasher: Shorten them by connecting jumpers on the solder side of the board as shown in Figure B. They go from pin 8 of IC's U33, U50, U68, and U81 to the ground feedthrough leading to C45.

4. Protect Fix

The bug: The protect line is floating, which allows noise pulses to set a memory board "protected" at the most inconvenient times.

The squasher: Connect a jumper wire on the solder side as shown in Figure B. It goes from the ground terminal of C 11 to pin 70 of the 100-pin bus connector J11.

5. DMA/Interrupt Unscramble

This fix has probably been included in your kit or preassembled board, but better check the connections just to make sure.

The bug: PINT (pin 73), PHOLD (pin 74), and PINTS (pin 26) got scrambled at an early stage in development and weren't noticed until too late.

The squasher: On the component side of the board, cut the trace leading to pin 73 on J11; on the solder side, cut the trace leading to pin 1 of U45, and also the second trace to the right of U64. (Refer to Figure B). Now connect three jumpers: From pin 73 of J11 to pin 1 of U45. From pin 28 of J11 to the feedthrough indicated (the one that was isolated by the cut on the component side). From the feedthrough directly below pin 1 of U45 to the feedthrough to the right of pin 3 of U64.

6 MWRITE Fix

The bug: If you want to operate with DMA devices which write into memory, such as discs, you; need to be able to generate the MWRITE pulse externally.

The squasher: You accomplish this by connecting the signals which generate MWRITE directly to the bus. On the solder side of the board, cut the trace which leads to pin 7 of U93. Now connect a jumper from the trace which has been isolated, to the feedthrough leading to pin 9 of U94. Now, still on the solder side, locate the feedthrough immediately below pin 1 of U92 and break the trace leading to it; do NOT break the trace leading to pin 1 of U92. Connect a jumper from that feedthrough to pin 13 of U107.

Further Remarks on D and E

The other major change between the D and E revision Sol boards involves reversing the order of the parallel input data lines as connected to the parallel connector J2. The schematic is correct for the D board, and the list of signals in the manual is correct for the E board. The change will simplify future connections to J2 by placing the POD lines adjacent to the PID lines; that way, you can create a bidirectional input/output bus with a simple jumpering scheme at the connector.

We're planning an adaptor connector to convert Rev. D J2 into Rev. E J2. It will have a 25-pin plug, a PC board which reverses the connections from pins 6 to 13, and a 25-pin socket connector. If you're developing a device which plugs into the Sol J2, reverse the order of pins 6 through 13 and use this adapter (PTC pt. 900011) to ensure that your plug-in device will be compatible with E revision Sol's.

7. Current Loop Fix

The bug: R23 and R24 should be connected to +12 volts instead of +5 volts.

The squasher: Break the large trace on the solder side of the board which leads to these two resistors. Still on the solder side, connect a jumper from the isolated end of R23 to the +12 volt feedthrough as shown. Be sure that you do not accidentally connect to the -12 volt feedthrough, which is slightly higher than the +12 volt one.

And a Bug in 5K(pre-Sol)

The bug: Our attention has been called to some problems with the integer function in BASIC-5 — the non-Sol version.

The squasher: Page 32 of the 5K BASIC manual (Software #2) should be changed to read as follows:

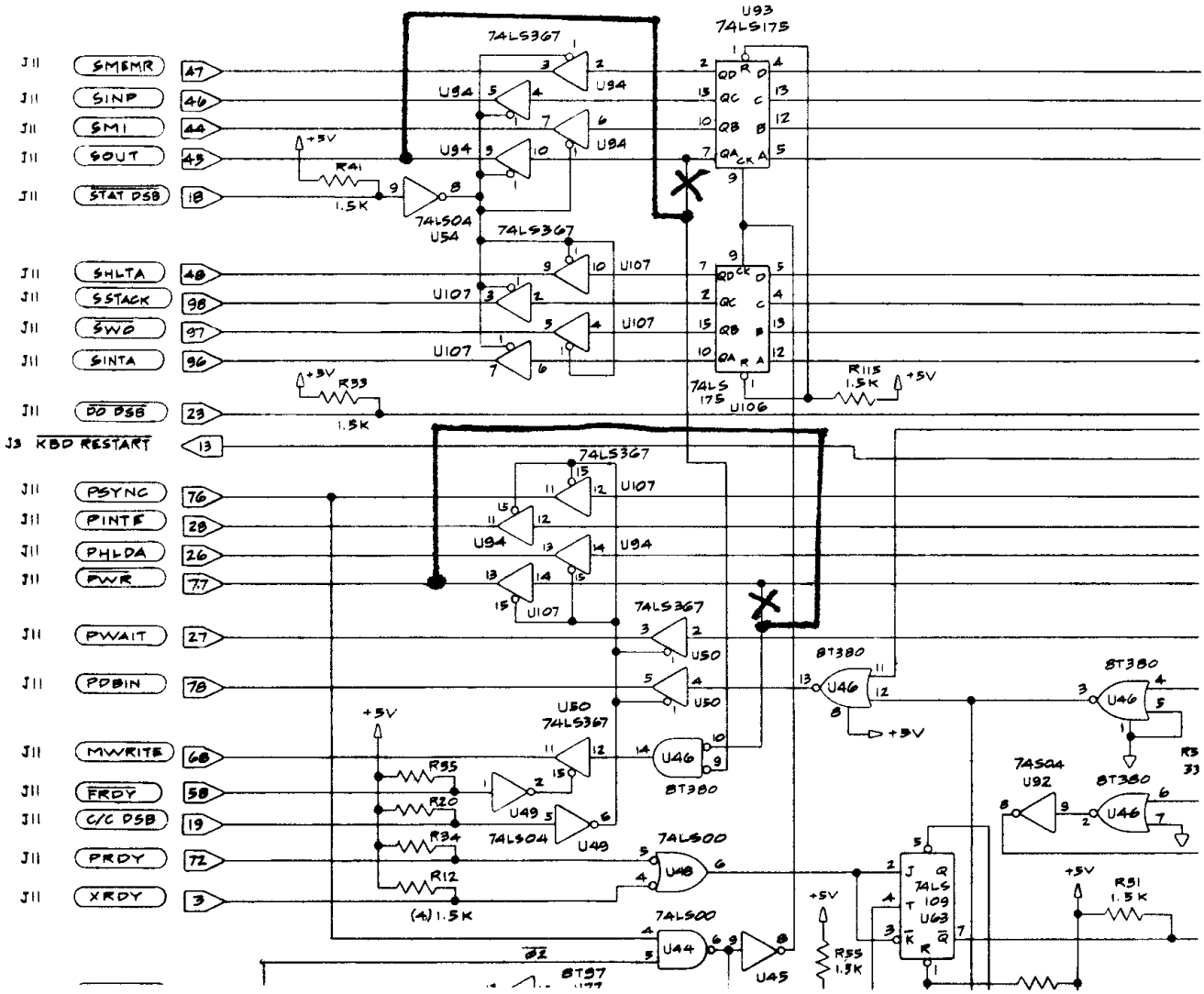
```
                    0000 *
                    0001 *
                    0002 *            BASIC-5 INTEGER FIX
                    0003
0BAA                0004          ORG     0BAAH
                    0005
0BAA 0A             0006 AINT     LDAX    B
0BAB D6 81          0007          SUI     129
0BAD 16 05          0008          MVI     D,FPSIZ
0BAF FA BF 0B       0009          JM      AINT3
0BB2 00             0010          NOP     .          SPACE FILLER
                    0011 *
                    0012 *  EXP > 0
                    0013 *
0BB3 D6 05          0014          SUI     FPNIB-1
0BB5 D0             0015          RNC
0BB6 57             0016          MOV     D,A        COUNT
0BB7 0B             0017          DCX     B
                    0018 *
0BB8 08             0019 AINT2    DCX     B
0BB9 0A             0020          LDAX    B
0BBA E6 F0          0021          ANI     360Q
0BBC 02             0022          STAX    B
0BBD 14             0023          INR     D
0BBE C8             0024          RZ
                    0025 *
0BBF AF             0026 AINT3    XRA     A
0BC0 02             0027          STAX    B
0BC1 14             0028          INR     D
0BC2 C2 B8 0B       0029          JNZ     AINT2
0BC5 C9             0030          RET
                    0031 *
                    0032 *
```

Your ALS-8 Applications Notes are on the Way!

After a seemingly interminable delay, the first batch of Application notes are really and truly in the mail to all you ALS-8 Users' Group members. Once the material is in your hot little hands (in a very fancy binder, no less), you'll be better able to appreciate the power and versatility of your ALS-8, and you should have some very happy hours of computing. If you haven't received your notes by the time you read this, please drop us a note right now, so we can track down whatever clerical or shipping errors crossed you up.

Our sincerest apologies for the delay.

Ninety-Ninety `Rule of Project Schedules:
The first ninety percent of the task takes ninety percent of the time and the last ten percent takes the other ninety percent.



SCHEMATIC, CPU & BUS, SOI		
SCALE:	APPROVED BY: ROBERT M MARSH	DRAWN BY LITO
DATE: 6-21-76		REVISED 9-23-76
PROCESSOR TECHNOLOGY		
REV. D		DRAWING NUMBER 1

MWRITE Fix

```
                        **        PROGRAM DEVELOPMENT SYSTEM

                             SOFTWARE TECHNOLOGY CORP.
CUTER (TM)   77-03-27          P.O. BOX 5260
COPYRIGHT (C) 1977            SAN MATEO, CA 94402


                        9999        COPY   CUTER1/1
                        0002 *
                        0003 *
                        0004 *
                        0005 *        CUTER(TM)
                        0006 *
                        0007 *                COPYRIGHT (C) 1977
                        0008 *                SOFTWARE TECHNOLOGY CORP.
                        0009 *                P.O. BOX 5260
                        0010 *                SAN MATEO, CA 94402
                        0011 *                (415) 349-8080
                        0012 *
                        0013 *    A L L    R I G H T S   R E S E R V E D ! ! !
                        0014 *
                        0015 *
                        0016 *        VERSION  1.3
                        0017 *                 77-03-27
                        0018 *
                        0019 *
                        0020 *  THIS PROGRAM IS DESIGNED TO BE A STANDALONE CUTS
                        0021 *  OPERATING SYSTEM.  CUTER IS DESIGNED TO BE READ IN FROM
                        0022 *  CASSETTE TAPE OR TO BE RESIDENT IN READ-ONLY-MEMORY.
                        0023 *  CUTER SUPPORTS VARIOUS DEVICES INCLUDING SERIAL,
                        0024 *  PARALLEL, THE PROCESSOR TECHNOLOGY VDM(TM) AND UP TO
                        0025 *  TWO CUTS TAPE DRIVES.
                        0026 *
                        0027 *  CUTER(TM) HAS BEEN WRITTEN SO AS TO BE COMPATIBLE WITH
                        0028 *  SOLOS(TM).  THE FOLLOWING KEYS ARE USED BY CUTER(TM)
                        0029 *  IN PLACE OF THE SPECIAL KEYS ON THE SOL KEYBOARD:
                        0030 *
                        0031 *     CURSOR UP       CTL-W
                        0032 *     CURSOR LEFT     CTL-A
                        0033 *     CURSOR RIGHT    CTL-S
                        0034 *     CURSOR DOWN     CTL-Z
                        0035 *     CURSOR HOME     CTL-N
                        0036 *     CLEAR SCREEN    CTL-K
                        0037 *     MODE            CTL-@
                        0038 *
                        0039 *
                        0040 *
                        0041 *
                        0042 *
                        0043 *   AUTO-STARTUP CODE
                        0044 *
C000 7F                 0045 START  MOV    A,A    SHOW THIS IS CUTER (SOLOS=00)
                        0046 *      THIS BYTE ALLOWS AUTOMATIC POWER ON ENTRY
                        0047 *      WHEN IN ROM SUPPORTING THIS HARDWARE FEATURE.
C001 C3 D7 C1           0048 INIT   JMP    STRTA  SYSTEM RESTART ENTRY POINT
                        0049 *
                        0050 *   THESE JUMP POINTS ARE PROVIDED TO ALLOW COMMON ENTRY
                        0051 * LOCATIONS FOR ALL VERSIONS OF CUTER.  THEY ARE USED
                        0052 * EXTENSIVELY BY CUTS SYSTEM PROGRAMS AND IT IS RECOMMENDED
                        0053 * THAT USER ROUTINES ACCESS CUTER ROUTINES THROUGH THESE
                        0054 * POINTS ONLY!
                        0055 *
C004 C3 18 C2           0056 RETRN  JMP    COMND  RETURN TO CUTER COMMAND PROCESSOR
C007 C3 DC C5           0057 FOPEN  JMP    BOPEN  CASSETTE OPEN FILE ENTRY
C00A C3 FF C5           0058 FCLOS  JMP    PCLOS  CASSETTE CLOSE FILE ENTRY
C00D C3 42 C6           0059 RDBYT  JMP    RTBYT  CASSETTE READ BYTE ENTRY
C010 C3 7F C6           0060 WRBYT  JMP    WTBYT  CASSETTE WRITE BYTE ENTRY
C013 C3 C7 C6           0061 RDBLK  JMP    RTAPE  CASSETTE READ BLOCK ENTRY
C016 C3 7B C7           0062 WRBLK  JMP    WTAPE  CASSETTE WRITE BLOCK ENTRY
                        0063 *
                        0064 *     SYSTEM I/O ENTRY POINTS
                        0065 *
                        0066 *  THESE FOUR ENTRY POINTS ARE USED TO EITHER INPUT
                        0067 *  OR OUTPUT TO CUTER PSUEDO PORTS.
                        0068 *  THESE PSUEDO PORTS ARE AS FOLLOWS:
                        0069 *
                        0070 *  PORT   INPUT              OUTPUT
                        0071 *  ----   -----------------  --------------------
                        0072 *   0     KEYBOARD INPUT     BUILT-IN VDM DRIVER
                        0073 *         ACTUAL PORT 3      PORT C8, MEMORY FROM CC00
                        0074 *   1     SERIAL PORT        SERIAL PORT
                        0075 *         ACTUAL PORT 1      ACTUAL PORT 1
                        0076 *   2     PARALLEL PORT      PARALLEL PORT
                        0077 *         ACTUAL PORT 2      ACTUAL PORT 2
                        0078 *   3     USER'S INPUT RTN   USER'S OUTPUT ROUTINE
```

```
                    0079 *
                    0080 *  STATUS FOR ACTUAL PORTS 1, 2 AND 3 IS VIA ACTUAL
                    0081 *  PORT 0.  THE BITS OF PORT ZERO ARE DEFINED AS FOLLOWS:
                    0082 *
                    0083 *   :     :     :     :     :     :---- : --- : --- :
                    0084 *   : TBE : RDA :     :     :     :PXDR : PDR : KDR :
                    0085 * BIT  7     6     5     4     3     2     1     0
                    0086 *
                    0087 *  WHERE:
                    0088 *    TBE    1=TRANSMITTER BUFFER EMPTY (SERIAL)
                    0089 *    RDA    1=READER DATA AVAILABLE (SERIAL)
                    0090 *    ----
                    0091 *    PXDR   0=PARALLEL EXTERNAL DEVICE READY
                    0092 *    ---
                    0093 *    PDR    0=PARALLEL DATA READY
                    0094 *    ---
                    0095 *    KDR    0=KEYBOARD DATA READY
                    0096 *
                    0097 *
                    0098 *
                    0099 *
                    0100 *  NOTE: SOUT AND SINP ARE "LDA" INSTRUCTIONS.
                    0101 *        THIS FACT IS USED TO ALLOW ACCESS TO THE
                    0102 *        BYTES "OPORT" AND "IPORT" DYNAMICALLY.
                    0103 *        THESE MUST REMAIN "LDA" INSTRUCTIONS!!!!!
                    0104 *
C019 3A 07 C8       0105 SOUT   LDA   OPORT   OUTPUT VIA STANDARD OUTPUT PSUEDO PORT
C01C C3 2E C0       0106 AOUT   JMP   OUTPR   OUTPUT VIA PSUEDO PORT SPECIFIED IN REG A
C01F 3A 06 C8       0107 SINP   LDA   IPORT   INPUT VIA STANDARD INPUT PSUEDO PORT
     C022           0108 AINP   EQU   $       INPUT VIA PSUEDO PORT SPECIFIED IN REG A
                    0109 * ----------END OF SYSTEM ENTRY POINTS----------
                    0110 *
                    0111 *
                    0112 * AINP CONTINUES HERE (IT COULD HAVE BEEN A "JMP" THOUGH)
C022 E5             0113        PUSH   H      SAVE HL FM ENTRY
C023 21 09 C3       0114        LXI    H,ITAB
                    0115 *
                    0116 *    THIS ROUTINE PROCESSES THE I/O REQUESTS
                    0117 *
C026 E6 03          0118 IOPRC  ANI    3      KEEP REGISTER "A" TO FOUR VALUES
C028 07             0119        RLC    .      COMPUTE ENTRY ADDRESS
C029 85             0120        ADD    L
C02A 6F             0121        MOV    L,A    WE HAVE ADDRESS
C02B C3 87 C2       0122        JMP    DISPT  DISPATCH TO IT
                    0123 *
                    0124 *
     C02E           0125 OUTPR  EQU    $      PROCESS OUTPUT REQUESTS
C02E E5             0126        PUSH   H      SAVE REGS
C02F 21 01 C3       0127        LXI    H,OTAB  POINT TO OUTPUT DISPATCH TABLE
C032 C3 26 C0       0128        JMP    IOPRC  DISPATCH FOR PROPER PSUEDO PORT
                    0129 *
                    0130 *
                    0131 *
                    0132 * CUTER SYSTEM I/O ROUTINES
                    0133 *
                    0134 *
                    0135 *    THIS ROUTINE IS A MODEL OF ALL INPUT ROUTINES WITHIN
                    0136 *  CUTER.  THE FIRST ROUTINE "KREA1" PERFORMS THE INPUT
                    0137 *  FROM THE STANDARD KEYBOARD ON PARALLEL PORT 3.
                    0138 *  ALL STANDARD INPUT DRIVERS RETURN EITHER THE CHARACTER
                    0139 *  WITH A NON-ZERO FLAG, OR JUST A ZERO FLAG INDICATING
                    0140 *  THAT NO CHARACTER IS AVAILABLE YET.  IT WILL BE THE
                    0141 *  RESPONSIBILITY OF THE USER TO LOOP WAITING FOR A
                    0142 *  CHARACTER, OR TO USE THE INPUT AS A STATUS REQUEST.
                    0143 *  WHEN A CHARACTER IS AVAILABLE, IT IS RETURNED IN REG A.
                    0144 *
                    0145 *  THE FOLLOWING KEYBOARD ROUTINE MAY BE USED AS A SAMPLE
                    0146 *  OF HOW TO WRITE A USER INPUT ROUTINE.
                    0147 *
                    0148 *         KEYBOARD INPUT ROUTINE
                    0149 *
     C035           0150 KREA1  EQU    $      KEYBOARD READ ROUTINE
C035 DB 00          0151        IN     STAPT  GET STATUS WORD
C037 2F             0152        CMA    .      INVERT IT FOR PROPER RETURN
C038 E6 01          0153        ANI    KDR    TEST NOT KEYBOARD DATA READY
C03A C8             0154        RZ     .      ZERO IF NO CHARACTER RECEIVED
                    0155 *
C03B DB 03          0156        IN     KDATA  GET CHARACTER
C03D C9             0157        RET    .      GO BACK WITH IT
                    0158 *
                    0159 *
                    0160 *
                    0161 *   SERIAL INPUT ROUTINE
                    0162 *
     C03E           0163 SREA1  EQU    $      SERIAL INPUT ROUTINE
```

```
C03E DB 00             0164        IN     STAPT  GET STATUS
C040 E6 40             0165        ANI    SDR    TEST FOR SERIAL DATA READY
C042 C8                0166        RZ     .      FLAGS ARE SET
                       0167 *
C043 DB 01             0168        IN     SDATA  GET DATA BYTE
                       0169 *  IT IS UP TO THE CALLER TO STRIP PARITY IF DESIRED
C045 C9                0170        RET    .      WE HAVE IT
                       0171 *
                       0172 *
                       0173 *   SERIAL DATA OUTPUT
                       0174 *
     C046              0175 SEROT  EQU    $      SERIAL OUTPUT ROUTINE
C046 DB 00             0176        IN     STAPT  GET STATUS
C048 17                0177        RAL    .      PUT HIGH BIT IN CARRY
C049 D2 46 C0          0178        JNC    SEROT  LOOP UNTIL TRANSMITTER BUFFER IS EMPTY
C04C 78                0179        MOV    A,B    GET THE CHARACTER BACK
C04D D3 01             0180        OUT    SDATA  SEND IT OUT
C04F C9                0181        RET    .      AND WE'RE DONE
                       0182 *
                       0183 *
                       0184 * PARALLEL DATA INPUT
     C050              0185 PARIT  EQU    $      GET CHAR FM PARALLEL PORT
C050 DB 00             0186        IN     STAPT  STATUS
C052 2F                0187        CMA    .      INVERT FOR PROPER RETURN
C053 E6 02             0188        ANI    PDR    IS DATA READY?
C055 C8                0189        RZ     .      NO--JUST EXIT
C056 DB 02             0190        IN     PDATA  YES--GET CHAR THEN
C058 C9                0191        RET    .      THEN EXIT
                       0192 *
                       0193 *
                       0194 *  PARALLEL DATA OUTPUT ROUTINE
     C059              0195 PAROT  EQU    $      OUTPUT CHAR TO PARALLEL PORT
C059 DB 00             0196        IN     STAPT  STATUS
C05B E6 04             0197        ANI    PXDR   IS EXTERNAL DEVICE READY?
C05D C2 59 C0          0198        JNZ    PAROT  NO--WAIT TIL IT IS
C060 78                0199        MOV    A,B    GET CHAR
C061 D3 02             0200        OUT    PDATA  SEND DATA NOW
C063 C9                0201        RET    .      DONE
                       0202 *
                       0203 *
                       0204 * USER DEFINED INPUT/OUTPUT ROUTINES
     C064              0205 ERRIT  EQU    $      USER INPUT ROUTINE
C064 E5                0206        PUSH   H      SAVE ORIG HL
C065 2A 00 C8          0207        LHLD   UIPRT  GET USER'S RTN ADDR
C068 C3 6F C0          0208        JMP    ERRO1  MERGE TO VERIFY THE ADDR
                       0209 *
     C06B              0210 ERROT  EQU    $      USER OUTPUT ROUTINE
C06B E5                0211        PUSH   H      SAVE ORIG HL
C06C 2A 02 C8          0212        LHLD   UOPRT  GET USER'S RTR ADDR
     C06F              0213 ERRO1  EQU    $      WE MERGE HERE TO VFY ADDR
C06F 7D                0214        MOV    A,L    ZERO=UNDEFINED
C070 B4                0215        ORA    H      IS IT?
C071 C2 8B C2          0216        JNZ    DISP1  NO--VALID--OFF TO IT
C074 C3 0F C2          0217        JMP    STRTD  RESET I/O PORTS AND BACK TO COMMAND MODE
                       0218 *
                       0219 *
                       0220 *
                       0221 *                  VIDEO DISPLAY ROUTINES
                       0222 *
                       0223 *
                       0224 *  THESE ROUTINES ALLOW FOR STANDARD VIDEO TERMINAL
                       0225 *  OPERATIONS.  ON ENTRY, THE CHARACTER FOR OUTPUT IS IN
                       0226 *  REGISTER B AND ALL REGISTERS ARE UNALTERED ON RETURN.
                       0227 *
                       0228 *
                       0229 *
     C077              0230 VDM01  EQU    $      VDM OUTPUT DRIVER
C077 E5                0231        PUSH   H      SAVE HL
C078 D5                0232        PUSH   D      SAVE DE
C079 C5                0233        PUSH   B
                       0234 *
                       0235 *  PROCESS ESC SEQUENCE IF ANY
                       0236 *
C07A 3A 0C C8          0237        LDA    ESCFL  GET ESCAPE FLAG
C07D B7                0238        ORA    A
C07E C2 87 C1          0239        JNZ    ESCS   IF NON ZERO GO PROCESS THE REST OF THE SEQUENCE
                       0240 *
C081 78                0241        MOV    A,B    GET CHAR
C082 E6 7F             0242        ANI    7FH    CLR HI BIT IN CASE
C084 47                0243        MOV    B,A    USE CHAR STRIPPED OF HI BIT FOR COMPATABILITY
C085 CA 9F C0          0244        JZ     GOBK   MAKE A QUICK EXIT FOR A NULL
                       0245 *
C088 21 E2 C2          0246        LXI    H,TBL
C08B CD A5 C0          0247        CALL   TSRCH  GO PROCESS
                       0248 *
```

```
     C08E              0249 GOBACK EQU    $      RESET CURSOR AND DELAY
C08E CD 44 C1          0250        CALL   VDADD  GET SCRN ADDR
C091 7E                0251        MOV    A,M    GET CHAR
C092 F6 80             0252        ORI    80H    INVERSE VIDEO
C094 77                0253        MOV    M,A    CURSOR IS NOW THERE
C095 2A 0A C8          0254        LHLD   SPEED-1  GET DELAY SPEED
C098 2C                0255        INR    L      MAKE IT DEFINITELY NON-ZERO
C099 AF                0256        XRA    A      DELAY ENDS WHEN H=ZERO
C09A 2B                0257 TIMER  DCX    H      LOOP FOR DELAY AMNT
C09B BC                0258        CMP    H      IS IT DONE YET
C09C C2 9A C0          0259        JNZ    TIMER  NO--KEEP DELAYING
C09F C1                0260 GOBK   POP    B
C0A0 D1                0261        POP    D      RESTORE ALL REGISTERS
C0A1 E1                0262        POP    H
C0A2 C9                0263        RET    .      EXIT FROM VDMOT
                       0264 *
                       0265 *
     C0A3              0266 NEXT   EQU    $      GO TO NEXT CHR
C0A3 23                0267        INX    H
C0A4 23                0268        INX    H
                       0269 *
                       0270 *  THIS ROUTINE SEARCHES FOR A MATCH OF THE CHAR IN "B"
                       0271 *  TO THE CHAR IN THE TBL POINTED TO BY HL.
                       0272 *
C0A5 7E                0273 TSRCH  MOV    A,M    GET CHR FROM TABLE
C0A6 B7                0274        ORA    A      SEE IF END OF TBL
C0A7 CA B7 C0          0275        JZ     CHAR   ZERO IS THE LAST
C0AA B8                0276        CMP    B      TEST THE CHR
C0AB 23                0277        INX    H      POINT FORWARD
C0AC C2 A3 C0          0278        JNZ    NEXT
C0AF E5                0279        PUSH   H      FOUND ONE...SAVE ADDRESS
C0B0 CD 5E C1          0280        CALL   CREM   REMOVE CURSOR
C0B3 E3                0281        XTHL   .      RESTORE ADDR OF CHAR ENTRY IN TBL
C0B4 C3 87 C2          0282        JMP    DISPT  DISPATCH FOR CURSOR CONTROL
                       0283 *
                       0284 *
     C0B7              0285 CHAR   EQU    $      WE HAVE A CHAR
C0B7 78                0286        MOV    A,B    GET CHARACTER
C0B8 FE 7F             0287        CPI    7FH    IS IT A DEL?
C0BA C8                0288        RZ     .      GO BACK IF SO
                       0289 *
                       0290 *
                       0291 *
C0BB CD 44 C1          0292 OCHAR  CALL   VDADD  GET SCREEN ADDRESS
C0BE 70                0293        MOV    M,B    PUT CHR ON SCREEN
C0BF 3A 08 C8          0294        LDA    NCHAR  GET CHARACTER POSITION
C0C2 FE 3F             0295        CPI    63     END OF LINE?
C0C4 DA E4 C0          0296        JC     OK
C0C7 3A 09 C8          0297        LDA    LINE
C0CA FE 0F             0298        CPI    15     END OF SCREEN?
C0CC C2 E4 C0          0299        JNZ    OK
                       0300 *
                       0301 *   END OF SCREEN...ROLL UP ONE LINE
                       0302 *
C0CF AF                0303 SCROLL XRA    A
C0D0 32 08 C8          0304        STA    NCHAR  BACK TO FIRST CHAR POSITION
C0D3 4F                0305 SROL   MOV    C,A
C0D4 CD 4B C1          0306        CALL   VDAD   CALCULATE LINE TO BE BLANKED
C0D7 AF                0307        XRA    A
C0D8 CD 22 C1          0308        CALL   CLIN1  CLEAR IT
C0DB 3A 0A C8          0309        LDA    BOT
C0DE 3C                0310        INR    A
C0DF E6 0F             0311        ANI    0FH
C0E1 C3 11 C1          0312        JMP    ERAS3
                       0313 *
                       0314 *   INCREMENT LINE COUNTER IF NECESSARY
                       0315 *
C0E4 3A 08 C8          0316 OK     LDA    NCHAR  GET CHR POSITION
C0E7 3C                0317        INR    A
C0E8 E6 3F             0318        ANI    3FH    MOD 64
C0EA 32 08 C8          0319        STA    NCHAR  STORE THE NEW
C0ED C0                0320        RNZ    .      MORE CHARS THIS LINE
     C0EE              0321 PDOWN  EQU    $      MOVE CURSOR DOWN ONE LINE
C0EE 3A 09 C8          0322        LDA    LINE   GET THE LINE COUNT
C0F1 3C                0323        INR    A
C0F2 E6 0F             0324 CURSC  ANI    0FH    MOD 15 INCREMENT
C0F4 32 09 C8          0325 CUR    STA    LINE   STORE THE NEW
C0F7 C9                0326        RET
                       0327 *
                       0328 *    ERASE SCREEN
                       0329 *
C0F8 21 00 CC          0330 PERSE  LXI    H,VDMEM  POINT TO SCREEN
C0FB 36 A0             0331        MVI    M,80H+' '  THIS IS THE CURSOR
                       0332 *
C0FD 23                0333        INX    H      NEXT CHAR
```

```
     C0F4              0334 ERAS1  EQU    $      LOOP TO CLR SCRN
C0FE 36 20             0335        MVI    M,' '  BLANK IT OUT
C100 23                0336        INX    H      NEXT SCRN LOC
C101 7C                0337        MOV    A,H    SEE IF DONE
C102 FE D0             0338        CPI    0D0H   DID IT GO ABOVE VDM
C104 DA FE C0          0339        JC     ERAS1  NO--MORE
C107 37                0340        STC    .      SAY WE WANT TO DROP THRU TO ERAS3
                       0341 *
     C108              0342 PHOME  EQU    $      RESET CURSOR TO HOME
C108 3E 00             0343        MVI    A,0    CLEAR, LEAVE CARRY AS IS
C10A 32 09 C8          0344        STA    LINE   ZERO LINE
C10D 32 08 C8          0345        STA    NCHAR  LEFT SIDE OF SCREEN
C110 D0                0346        RNC    .      THIS IS JUST A HOME OPERATION
                       0347 *
C111 D3 C8             0348 ERAS3  OUT    DSTAT  RESET SCROLL PARAMETERS
C113 32 0A C8          0349        STA    BOT    BEGINNING OF TEXT OFFSET
C116 C9                0350        RET
                       0351 *
                       0352 *
     C117              0353 CLIN2  EQU    $      HERE TO SEE IF VDM OUTPUT
C117 3A 07 C8          0354        LDA    OPORT  GET CRNT OUTPUT PORT
C11A B7                0355        ORA    A
C11B C0                0356        RNZ    .      NOT VDM--DONE THEN
C11C CD 44 C1          0357 CLINE  CALL   VDADD  GET CURRENT SCREEN ADDRESS
C11F 3A 08 C8          0358        LDA    NCHAR  CURRENT CURSOR POSITION
C122 FE 40             0359 CLIN1  CPI    64     NO MORE THAN 63
C124 D0                0360        RNC    .      ALL DONE
C125 36 20             0361        MVI    M,' '  ALL SPACED OUT
C127 23                0362        INX    H
C128 3C                0363        INR    A
C129 C3 22 C1          0364        JMP    CLIN1  LOOP TO END OF LINE
                       0365 *
                       0366 *
                       0367 *  ROUTINE TO MOVE THE CURSOR UP ONE LINE
                       0368 *
C12C 3A 09 C8          0369 PUP    LDA    LINE   GET LINE COUNT
C12F 3D                0370        DCR    A
C130 C3 F2 C0          0371        JMP    CURSC  MERGE
                       0372 *
                       0373 *  MOVE CURSOR LEFT ONE POSITION
                       0374 *
C133 3A 08 C8          0375 PLEFT  LDA    NCHAR
C136 3D                0376        DCR    A
     C137              0377 PCUR   EQU    $      TAKE CARE OF CURSOR SAME LINE
C137 E6 3F             0378        ANI    03FH   LET CURSOR WRAP AROUND
C139 32 08 C8          0379        STA    NCHAR  UPDATED CURSOR
C13C C9                0380        RET
                       0381 *
                       0382 *     CURSOR RIGHT ONE POSITION
                       0383 *
C13D 3A 08 C8          0384 PRIT   LDA    NCHAR
C140 3C                0385        INR    A
C141 C3 37 C1          0386        JMP    PCUR
                       0387 *
                       0388 *   ROUTINE TO CALCULATE SCREEN ADDRESS
                       0389 *
                       0390 *   ENTRY AT:    RETURNS:
                       0391 *
                       0392 *         VDADD  CURRENT SCREEN ADDRESS
                       0393 *         VDAD2  ADDRESS OF CURRENT LINE, CHAR 'C'
                       0394 *         VDAD   LINE 'A', CHARACTER POSITION 'C'
                       0395
C144 3A 08 C8          0396 VDADD  LDA    NCHAR  GET CHARACTER POSITION
C147 4F                0397        MOV    C,A    'C' KEEPS IT
C148 3A 09 C8          0398 VDAD2  LDA    LINE   LINE POSITION
C14B 6F                0399 VDAD   MOV    L,A    INTO 'L'
C14C 3A 0A C8          0400        LDA    BOT    GET TEXT OFFSET
C14F 85                0401        ADD    L      ADD IT TO THE LINE POSITION
C150 0F                0402        RRC    .      TIMES TWO
C151 0F                0403        RRC    .      MAKES FOUR
C152 6F                0404        MOV    L,A    L HAS IT
C153 E6 03             0405        ANI    3      MOD THREE FOR LATER
C155 C6 CC             0406        ADI    <VDMEM  LOW SCREEN OFFSET
C157 67                0407        MOV    H,A    NOW H IS DONE
C158 7D                0408        MOV    A,L    TWIST L'S ARM
C159 E6 C0             0409        ANI    0C0H
C15B 81                0410        ADD    C
C15C 6F                0411        MOV    L,A
C15D C9                0412        RET    .      H & L ARE NOW PERVERTED
                       0413 *
                       0414 *    ROUTINE TO REMOVE CURSOR
                       0415 *
C15E CD 44 C1          0416 CREM   CALL   VDADD  GET CURRENT SCREEN ADDRESS
C161 7E                0417        MOV    A,M
C162 E6 7F             0418        ANI    7FH    STRIP OFF THE CURSOR
```

```
C164 77                0419        MOV    M,A
C165 C9                0420        RET
                       0421 *
                       0422 *     ROUTINE TO BACKSPACE
                       0423 *
C166 CD 33 C1          0424 PBACK  CALL   PLEFT
C169 CD 44 C1          0425        CALL   VDADD  GET SCREEN ADDRESS
C16C 36 20             0426        MVI    M,' '  PUT A BLANK THERE
C16E C9                0427        RET
                       0428 *
                       0429 *     ROUTINE TO PROCESS A CARRIAGE RETURN
                       0430 *
C16F CD 1C C1          0431 PCR    CALL   CLINE  CLEAR FROM CURRENT CURSOR TO END OF LINE
                       0432 *  NOTE THAT A COMES BACK=64 WHICH WILL BE CLEARED AT PCUR
C172 C3 37 C1          0433        JMP    PCUR   AND STORE THE NEW VALUE
                       0434 *
                       0435 *   ROUTINE TO PROCESS LINEFEED
                       0436 *
C175 3A 09 C8          0437 PLF    LDA    LINE   GET LINE COURT
C178 3C                0438        INR    A      NEXT LINE
C179 E6 0F             0439        ANI    15     SEE IF IT WRAPPED AROUND
C17B C2 F4 C0          0440        JNZ    CUR    IT DID NOT--NO SCROLL
                       0441 *
C17E C3 D3 C0          0442        JMP    SROL   SCROLL ONE LINE--CURSOR SOME POSITION
                       0443 *
                       0444 *     SET ESCAPE PROCESS FLAG
                       0445 *
C181 3E FF             0446 PESC   MVI    A,-1
C183 32 0C C8          0447        STA    ESCFL  SET FLAG
C186 C9                0448        RET
                       0449 *
                       0450 *       PROCESS ESCAPE SEQUENCE
                       0451 *
C187 CD 5E C1          0452 ESCS   CALL   CREM   REMOVE CURSOR
C18A CD 90 C1          0453        CALL   ESCSP  PROCESS THE CHARACTER
C18D C3 8E C0          0454        JMP    GOBACK
                       0455 *
C190 3A 0C C8          0456 ESCSP  LDA    ESCFL  GET ESCAPE FLAG
C193 FE FF             0457        CPI    -1     TEST FLAG
C195 CA B8 C1          0458        JZ     SECOND
                       0459 *
                       0460 *  PROCESS THIRD CHR OF ESC SEQUENCE
                       0461 *
C198 21 0C C8          0462        LXI    H,ESCFL
C19B 36 00             0463        MVI    M,0
C19D FE 02             0464        CPI    2
C19F DA B0 C1          0465        JC     SETX   SET X
C1A2 CA B4 C1          0466        JZ     SETY   SET Y
C1A5 FE 08             0467        CPI    8      SPECIAL SET SPEED
C1A7 CA 94 C5          0468        JZ     STSPD  YES--SET
C1AA FE 09             0469        CPI    9
C1AC DA BB C0          0470        JC     OCHAR  PUT IT ON THE SCREEN
C1AF C0                0471        RNZ
                       0472 *
                       0473 *  TAB ABSOLUTE TO VALUE IN REG B
                       0474 *
C1B0 78                0475 SETX   MOV    A,B
C1B1 C3 37 C1          0476        JMP    PCUR
                       0477 *
                       0478 *  SET CURSOR TO LINE "B"
                       0479 *
C1B4 78                0480 SETY   MOV    A,B
C1B5 C3 F2 C0          0481        JMP    CURSC
                       0482 *
                       0483 *
                       0484 *   PROCESS SECOND CHR OF ESC SEQUENCE
                       0485 *
C1B8 78                0486 SECOND MOV    A,B
C1B9 FE 03             0487        CPI    3
C1BB CA CE C1          0488        JZ     CURET
C1BE FE 04             0489        CPI    4
C1C0 C2 CA C1          0490        JNZ    ARET2
                       0491 *
C1C3 44                0492 ARET   MOV    B,H
C1C4 4D                0493        MOV    C,L    PRESENT SCREEN ADDRESS TO BC FOR RETURN
C1C5 E1                0494 ARET1  POP    H      RETURN ADDRESS
C1C6 D1                0495        POP    D      OLD B
C1C7 C5                0496        PUSH   B
C1C8 E5                0497        PUSH   H
C1C9 AF                0498        XRA    A
C1CA 32 0C C8          0499 ARET2  STA    ESCFL
C1CD C9                0500        RET
                       0501 *
                       0502 *
                       0503 *     RETURN PRESENT SCREEN PARAMETERS IN BC
```

```
                        0504 *
C1CE 21 08 C8           0505 CURET  LXI    H,NCHAR
C1D1 46                 0506        MOV    B,M    CHARACTER POSITION
C1D2 23                 0507        INX    H
C1D3 4E                 0508        MOV    C,M    LINE POSITION
C1D4 C3 C5 C1           0509        JMP    ARET1
                        0510 *
                        0511 *
                        0512 *
                        0513 *               START UP SYSTEM
                        0514 *
                        0515 *   CLEAR SCREEN AND THE FIRST 256 BYTES OF GLOBAL RAM
                        0516 *  THEN ENTER THE COMMAND MODE.
                        0517 *
C1D7 AF                 0518 STRTA  XRA    A
C1D8 4F                 0519        MOV    C,A
C1D9 21 04 C8           0520        LXI    H,DFLTS  CLEAR AFTER USER PORT ADDRESSES
                        0521 *
C1DC 77                 0522 CLERA  MOV    M,A
C1DD 23                 0523        INX    H
C1DE 0C                 0524        INR    C
C1DF C2 DC C1           0525        JNZ    CLERA
                        0526 *
                        0527 * DETERMINE THE DEFAULT PORTS
                        0528 *     THIS COULD BECOME "MVI A,XX" FOR YOUR SPECIFIC PORTS
C1E2 DB FF              0529        IN     SENSE  GET SWITCHES
                        0530 *
C1E4 47                 0531        MOV    B,A    SAVE IT
C1E5 E6 03              0532        ANI    3      MAKE IT A VALID PORT
C1E7 32 05 C8           0533        STA    DFLTS+1  SET DEFAULT OUTPUT PORT
C1EA B7                 0534        ORA    A      SEE IF THIS THE VDM
C1EB C2 F4 C1           0535        JNZ    STRTB  NO--DO NOT RESET VDM
C1EE 31 FF CB           0536        LXI    SP,SYSTP SET UP THE STACK FOR CALL
C1F1 CD F8 C0           0537        CALL   PERSE  (REG A ASSUMED TO COME BACK ZERO)
     C1F4               0538 STRTB  EQU    $      FINISH OFF THIS PORT THEN DO NEXT
C1F4 21 00 00           0539        LXI    H,0    USE FOR CLEARING USER ADDRESSES
C1F7 FE 03              0540        CPI    3      IS IT A USER PORT
C1F9 CA FF C1           0541        JZ     STRTC  YES-- DO NOT CLEAR IT
C1FC 22 02 C8           0542        SHLD   UOPRT  NO--CLEAR ADDR
     C1FF               0543 STRTC  EQU    $      OUTPUT PORT ALL SET
C1FF 78                 0544        MOV    A,B    FM SENSE SWITCHES
C200 1F                 0545        RAR    .
C201 1F                 0546        RAR    .      NEXT 2 SITS ARE INPUT PORT
C202 E6 03              0547        ANI    3      VALID PORT
C204 32 04 C8           0548        STA    DFLTS  THIS IS DEFAULT INPUT PORT
C207 FE 03              0549        CPI    3      IS THIS ONE A USER PORT
C209 CA 0F C2           0550        JZ     STRTD  YES--DO NOT CLEAR IT THEN
C20C 22 00 C8           0551        SHLD   UIPRT  NO--FORCE USER ADDRESS ZERO
     C20F               0552 STRTD  EQU    $      1ST TIME INITIALIZATION ALL DONE NOW
C20F 2A 04 C8           0553        LHLD   DFLTS  PICK UP DEFAULT PORTS
C212 22 06 C8           0554        SHLD   IPORT  FORCE PORTS TO DEFAULT
     C215               0555 COMN1  EQU    $      HERE TO TURN OFF TAPES, THEN COMMAND MODE
C215 AF                 0556        XRA    A
C216 D3 FA              0557        OUT    TAPPT  BE SURE TAPES ARE OFF
                        0558 *
                        0559 *
                        0560 *
                        0561 *            =--  COMMAND MODE  --=
                        0562 *
                        0563 *
                        0564 *   THIS ROUTINE GETS AND PROCESSES COMMANDS
                        0565 *
C218 31 FF CB           0566 COMND  LXI    SP,SYSTP  SET STACK POINTER
C21B CD 3A C3           0567        CALL   PROMPT  PUT PROMPT ON SCREEN
C21E CD 27 C2           0568        CALL   GCLIO  INIT TO GET COMMAND LINE
C221 CD 6A C2           0569        CALL   COPRC  PROCESS THE LINE
C224 C3 18 C2           0570        JMP    COMND  OVER AND OVER
                        0571 *
                        0572 *
                        0573 *
                        0574 *   THIS ROUTINE READS A COMMAND LINE FROM THE SYSTEM
                        0575 *  KEYBOARD
                        0576 *
                        0577 *  C/R   TERMINATES THE SEQUENCE ERASING ALL CHARS TO THE
                        0578 *        RIGHT OF THE CURSOR
                        0579 *  L/F   TERMINATES THE SEQUENCE
                        0580 *  ESC   RESETS TO COMMAND MODE.
                        0581 *
     C227               0582 GCLI0  EQU    $      HERE TO INIT FOR GCLIN
C227 21 63 CA           0583        LXI    H,INLIN-1  PT TO CHAR IN FRONT OF INPUT RFR
C22A 36 07              0584        MVI    M,7    MAKE SURE IT IS "BELL" TO KEEP FM DEL'ING TOO FAR
C22C 23                 0585        INX    H      NOW PT TO INPUT BFR
C22D 22 0E C8           0586        SHLD   INPTR  SAVE AS STARTING PTR
C230 3E 50              0587        MVI    A,80   NUMBER OF CHARS IN LINE (MAX)
     C232               0588 GCLI1  EQU    $      LOOP TO BLANK OUT LINE BFR
```

```
C232 36 20             0589        MVI    M,     BLANKS
C234 23                0590        INX    H      NEXT CHAR
C235 3D                0591        DCR    A      FOR THIS COUNT
C236 C2 32 C2          0592        JNZ    GCLI1  ENTIRE LINE
C239 CD 1F C0          0593 GCLIN  CALL   SINP   READ INPUT DEVICE
C23C CA 39 C2          0594        JZ     GCLIN
C23F E6 7F             0595        ANI    7FH    MAKE SURE NO X'80' BIT DURING CMND MODE
C241 CA 0F C2          0596        JZ     STRTD  IF EITHER MODE (OR CTL-@)
C244 47                0597        MOV    B,A
C245 FE 0D             0598        CPI    CR IS IT CR?
C247 CA 17 C1          0599        JZ     CLIN2  YES--TERMINATE LINE HERE (CLR IF VDM)
C24A FE 0A             0600        CPI    LF     IS IT A LINEFEED
C24C C8                0601        RZ     .      YES--TERMINATE LINE AS IS
C24D 2A 0E C8          0602        LHLD   INPTR  CRNT LINE PTR
C250 FE 7F             0603        CPI    7FH    DELETE CHR?
C252 C2 5F C2          0604        JNZ    GCLI2  NO--OK
C255 06 5F             0605        MVI    B,BACKS  REPLACE IT
C257 2B                0606        DCX    H      BACK LINE PTR UP TOO
C258 3E 07             0607        MVI    A,'G'-40H  SEE IF A BELL
C25A BE                0608        CMP    M      IS IT?
C25B C2 61 C2          0609        JNZ    GCLI3  NO--OK
C25E 47                0610        MOV    B,A    YES--RING THE BELL THEN
     C25F              0611 GCLI2  EQU    $      STORE CHAR IN INPUT AREA
C25F 70                0612        MOV    M,B    PLACE CHAR INTO LINE
C260 23                0613        INX    H      NEXT CHAR
     C261              0614 GCLI3  EQU    $      SAVE NEW LINE PTR
C261 22 0E C8          0615        SHLD   INPTR  SAVE PTR
                       0616 *
C264 CD 19 C0          0617 CONT   CALL   SOUT
C267 C3 39 C2          0618        JMP    GCLIN
                       0619 *
                       0620 *
                       0621 *
                       0622 *
                       0623 *      FIND AND PROCESS COMMAND
                       0624 *
     C26A              0625 COPRC  EQU    $      PROCESS THIS COMMAND LINE
C26A CD AA C2          0626        CALL   STUP   SETUP TO PROCESS INPUT LINE
C26D EB                0627        XCHG   .      DE=ADDR
C26E 21 00 C0          0628        LXI    H,START  PREP SO THAT HL WILL PT TO CUTER LATER
C271 E5                0629        PUSH   H      PLACE PTR TO CUTER ON STACK FOR LATER DISPT
C272 CD 6C C3          0630        CALL   SCHR   SCAN PAST BLANKS
C275 CA 6B C4          0631        JZ     ERR1   NO COMMAND?
C278 EB                0632        XCHG   .      HL HAS FIRST CHR
C279 11 BD C2          0633        LXI    D,COMTAB  POINT TO COMMAND TABLE
C27C CD 91 C2          0634        CALL   FDCOM  SEE IF IN PRIMARY TABLE
C27F CC 8E C2          0635        CZ     FDCOU  TRY CUSTOM ONLY IF NOT PRIMARY COMMAND
     C282              0636 DISP0  EQU    $      HERE TO EITHER DISPATCH OR DO ERROR
C282 CA 6C C4          0637        JZ     ERR2   NOT IN EITHER TABLE
C285 13                0638        INX    D      PT DE TO ADDR OF RTN
C286 EB                0639        XCHG   .      HL=ADDR OF ADDR OF RTN
                       0640 * **** DROP THRU TO DISPT ***
                       0641 *
                       0642 * THIS ROUTINE DISPTACHES TO THE ADDR AT CONTENTS OF HL.
                       0643 * HL ARE RESTORED PRIOR TO GOING TO ROUTINE.
                       0644 *
     C287              0645 DISPT  EQU    $      DISPATCH
C287 7E                0646        MOV    A,M    LOW BYTE
C288 23                0647        INX    H
C289 66                0648        MOV    H,M    HI BYTE
C28A 6F                0649        MOV    L,A    AND LO, HL NOW COMPLETE
     C28B              0650 DISP1  EQU    $      HERE TO GO OFF TO HL DIRECTLY
C28B E3                0651        XTHL   .      HL RESTORED AND ADDR ON STACK
C28C 7D                0652        MOV    A,L    ALWAYS PASS L IN "A" (PRIMARILY FOR SET'S)
C28D C9                0653        RET    .      OFF TO ROUTINE
                       0654 *
                       0655 *
                       0656 *
                       0657 *   THIS ROUTINE SEARCHES THROUGH A TABLE, POINTED TO
                       0658 *  BY 'DE', FOR A DOUBLE CHARACTER MATCH OF THE 'HL'
                       0659 *  MEMORY CONTENT.  IF NO MATCH IS FOUND THE SCAN ENDS
                       0660 *  WITH THE ZERO FLAG SET, ELSE NON-ZERO SET.
                       0661 *
     C28E              0662 FDCOU  EQU    $      HERE TO SCAN CUSTOM TABLE
C28E 11 3C C8          0663        LXI    D,CUTAB  PT TO CUSTOM RTN TBL
C291 1A                0664 FDCOM  LDAX   D
C292 B7                0665        ORA    A      TEST FOR TABLE END
C293 C8                0666        RZ     .      NOT FOUND POST THAT AND RETURN
C294 E5                0667        PUSH   H      SAVE START OF SCAN ADDRESS
C295 BE                0668        CMP    M      TEST FIRST CHR
C296 13                0669        INX    D
C297 C2 A3 C2          0670        JNZ    NCOM
                       0671 *
C29A 23                0672        INX    H
C29B 1A                0673        LDAX   D
```

```
C29C BE                 0674         CMP    M       NOW SECOND CHARACTER
C29D C2 A3 C2           0675         JNZ    NCOM    GOODNESS
                        0676 *
C2A0 E1                 0677         POP    H       RETURN HL TO PT TO CHAR START
C2A1 B7                 0678         ORA    A       FORCE TO NON-ZERO FLAG
C2A2 C9                 0679         RET    .       LET CALLER KNOW
                        0680 *
                        0681 *
C2A3 13                 0682 NCOM    INX    D       G0 TO NEXT ENTRY
C2A4 13                 0683         INX    D
C2A5 13                 0684         INX    D
C2A6 E1                 0685         POP    H       GET BACK ORIGINAL ADDRESS
C2A7 C3 91 C2           0686         JMP    FDCOM   CONTINUE SEARCH
                        0687 *
                        0688 *
                        0689 * SET UP TO PROCESS AN INPUT LINE
     C2AA               0690 STUP    EQU    $       PREPARE WHETHER VDM OR NOT
C2AA 21 64 CA           0691         LXI    H,INLIN ASSUME NON-VDM INPUT
C2AD 22 0E C8           0692         SHLD   INPTR   ALSO RESET PTR FOR NOW
C2B0 3A 07 C8           0693         LDA    OPORT   SEE IF IT IS VDM
C2B3 B7                 0694         ORA    A       IS IT THE VDM PORT
C2B4 C0                 0695         RNZ    .       NO--HL ARE SET PROPERLY
C2B5 CD 5E C1           0696         CALL   CREM    REMOVE CURSOR
C2B8 0E 01              0697         MVI    C,1     GET VDM ADDR FM POSITION ONE
C2BA C3 48 C1           0698         JMP    VDAD2   GET SCRN ADDR
                        0699 *
                        0700 *           COMMAND TABLE
                        0701 *
                        0702 *  THIS TABLE DESCRIBES THE VALID COMMANDS FOR CUTER
                        0703 *
     C2BD               0704 COMTAB  EQU    $       START OF KNOWN COMMANDS
C2BD 44 55              0705         ASC    'DU'    DUMP
C2BF AD C3              0706         DW     DUMP
C2C1 45 4E              0707         ASC    'EN'    ENTR
C2C3 14 C4              0708         DW     ENTER
C2C5 45 58              0709         ASC    'EX'    EXEC
C2C7 49 C4              0710         DW     EXEC
C2C9 47 45              0711         ASC    'GE'    GET
C2CB A1 C4              0712         DW     TLOAD
C2CD 53 41              0713         ASC    'SA'    SAVE
C2CF E0 C4              0714         DW     TSAVE
C2D1 58 45              0715         ASC    'XE'    XEQ
C2D3 A0 C4              0716         DW     TXEQ
C2D5 43 41              0717         ASC    'CA'    CAT
C2D7 27 C5              0718         DW     TLIST
C2D9 53 45              0719         ASC    'SE'    SET COMMAND
C2DB 76 C5              0720         DW     SET
C2DD 43 55              0721         ASC    'CU'    CUSTOM COMMAND ENTER/CLEAR
C2DF B9 C5              0722         DW     CUSET
C2E1 00                 0723         DB     0       END OF TABLE MARK
                        0724 *
                        0725 *
                        0726 *              DISPLAY DRIVER COMMAND TABLE
                        0727 *
                        0728 *     THIS TABLE DEFINES THE CHARACTERS FOR SPECIAL
                        0729 *  PROCESSING.  IF THE CHARACTER IS NOT IN THE TARLP IT
                        0730 *  GOES TO THE SCREEN.
                        0731 *
C2E2 0B                 0732 TBL     DB     CLEAR   SCREEN
C2E3 F8 C0              0733         DW     PERSE
C2E5 17                 0734         DB     UP      CURSOR
C2E6 2C C1              0735         DW     PUP
C2E8 1A                 0736         DB     DOWN    "
C2E9 EE C0              0777         DW     PDOWN
C2EB 01                 0738         DB     LEFT    "
C2EC 33 C1              0739         DW     PLEFT
C2EE 13                 0740         DB     RIGHT   "
C2EF 3D C1              0741         DW     PRIT
C2F1 0E                 0742         DB     HOME    "
C2F2 08 C1              0743         DW     PHOME
C2F4 0D                 0744         DB     CR      CARRIAGE RETURN
C2F5 6F C1              0745         DW     PCR
C2F7 0A                 0746         DB     LF      LINE FEED
C2F8 75 C1              0747         DW     PLF
C2FA 5F                 0748         DB     BACKS   BACK SPACE
C2FB 66 C1              0749         DW     PBACK
C2FD 1B                 0750         DB     ESC     ESCAPE KEY
C2FE 81 C1              0751         DW     PESC
C300 00                 0752         DB     0       END OF TABLE
                        0753 *
                        0754 *   OUTPUT DEVICE TABLE
                        0755 *
C301 77 C0              0756 OTAB    DW     VDM01   VDM DRIVER
C303 46 C0              0757         DW     SEROT   SERIAL OUTPUT
C305 59 C0              0758         DW     PAROT   PARALLEL OUTPUT
```

```
C307 6B C0              0759         DW     ERROT  ERROR OR USER DRIVER HANDLER
                        0760 *
                        0761 *    INPUT DEVICE TABLE
                        0762 *
C309 35 C0              0763 ITAB    DW     KREA1  KEYBOARD INPUT
C30B 3E C0              0764         DW     SREA1  SERIAL INPUT
C30D 50 C0              0765         DW     PARIT  PARALLEL INPUT
C30F 64 C0              0766         DW     ERRIT  ERROR OR USER DRIVER HANDLER
                        0767 *
                        0768 *
                        0769 *       SECONDARY COMMAND TABLE FOR SET COMMAND
                        0770 *
C311 54 41              0771 SETAB   ASC    'TA'   SET TAPE SPEED
C313 8A C5              0772         DW     TASPD
C315 53 3D              0773         ASC    'S='   SET DISPLAY SPEED
C317 95 C5              0774         DW     DISPD
C319 49 3D              0775         ASC    'I='   SET INPUT PORT
C31B 99 C5              0776         DW     SETIN
C31D 4F 3D              0777         ASC    'O='   SET OUTPUT PORT
C31F 9D C5              0778         DW     SETOT
C321 43 49              0779         ASC    'CI'   SET CUSTOM DRIVER ADDRESS
C323 A1 C5              0780         DW     SETCI
C325 43 4F              0781         ASC    'CO'   SET CUSTOM OUTPUT DRIVER ADDRESS
C327 A5 C5              0782         DW     SETCO
C329 58 45              0783         ASC    'XE'   SET HEADER XEQ ADDRESS
C32B AD C5              0784         DW     SETXQ
C32D 54 59              0785         ASC    'TY'   SET HEADER TYPE
C32F A9 C5              0786         DW     SETTY
C331 4E 3D              0787         ASC    'N='   SET NUMBER OF NULLS
C333 B1 C5              0788         DW     SETNU
C335 43 52              0789         ASC    'CR'   SET CRC (NORMAL OR IGNORE CRC ERRORS)
C337 B5 C5              0790         DW     SETCR
C339 00                 0791         DB     0      END OF TABLE MARK
                        0792 * -*-
                        9999         COPY  CUTER2/1                              2 OF 3 ****
                        0793 *
                        0794 *
                        0795 *       OUTPUT A CRLF FOLLOWED BY A PROMPT
                        0796 *
C33A CD 42 C3           0797 PROMPT CALL   CRLF
C33D 06 3E              0798         MVI    B,'>'  THE PROMPT
C33F C3 19 C0           0799         JMP    SOUT   PUT IT ON THE SCREEN
                        0800 *
C342 06 0A              0801 CRLF    MVI    B,LF   LINE FEED
C344 CD 19 C0           0802         CALL   SOUT
C347 06 0D              0803         MVI    B,CR   CARRIAGE RETURN
C349 CD 19 C0           0804         CALL   SOUT
C34C 3A 10 C8           0805         LDA    NUCNT  GET COUNT OF NULLS TO OUTPUT
C34F 4F                 0806         MOV    C,A    SAVE COUNT IN C
C350 0D                 0807 NULOT   DCR    C
C351 F8                 0808         RM     .      COUNTED DOWN PAST ZERO (MAX COUNT IS X'7F')
C352 AF                 0809         XRA    A      HERE IS THE NULL
C353 CD 10 C4           0810         CALL   OUTH   OUTPUT IT
C356 C3 50 C3           0811         JMP    NULOT  LOOP FOR NUMBER OF NULLS
                        0812 *
                        0813 *
                        0814 *  SCAN OVER UP TO 12 CHARACTERS LOOKING FOR A BLANK
                        0815 *
C359 DE 0C              0816 SBLK    MVI    C,12   MAXIMUM COMMAND STRING
C35B 1A                 0817 SBLK1   LDAX   D
C35C FE 20              0818         CPI    BLANK
C35E CA 6C C3           0819         JZ     SCHR   GOT A BLANK NOW SCAN PAST IT
C361 13                 0820         INX    D
C362 FE 3D              0821         CPI    '='    A EQUAL WILL ALSO STOP US (AT NEXT CHAR)
C364 CA 6C C3           0822         JZ     SCHR   FOUND, DE PT TO NEXT CHAR
C367 0D                 0823         DCR    C      NO MORE THAN TWELVE
C368 C2 5B C3           0824         JNZ    SBLK1
C36B C9                 0825         RET    .      GO BACK WITH ZERO FLAG SET
                        0826 *
                        0827 *
                        0828 *  SCAN PAST UP TO 10 BLANK POSITIONS LOOKING FOR
                        0829 * A NON BLANK CHARACTER.
                        0830 *
C36C 0E 0A              0831 SCHR    MVI    C,10   SCAN TO FIRST NON BLANK CHR WITHIN 10
C36E 1A                 0832 SCHR1   LDAX   D      GET NEXT CHARACTER
C36F FE 20              0833         CPI    SPACE
C371 C0                 0834         RNZ    .      WE'RE PAST THEM
C372 13                 0835         INX    D      NEXT SCAN ADDRESS
C373 0D                 0836         DCR    C
C374 C8                 0837         RZ     .      COMMAND ERROR
C375 C3 6E C3           0838         JMP    SCHR1  KEEP LOOPING
                        0839 *
                        0840 *  THIS ROUTINE SCANS OVER CHARACTERS, PAST BLANKS AND
                        0841 * CONVERTS THE FOLLOWING ADDRESS TO HEX.  ERRORS RETURN TO
                        0842 * THE ERROR HANDLER.
```

```
                        0843 *
C378 CD 59 C3           0844 SCONV  CALL   SBLK
C37B CA 6B C4           0845        JZ     ERR1
                        0846 *
                        0847 *  THIS ROUTINE CONVERTS ASCII DIGITS INTO BINARY FOLLOWING
                        0848 * A STANDARD HEX CONVERSION.  THE SCAN STOPS WHEN AN ASCII
                        0849 * SPACE IS ENCOUNTERED.  PARAMETER ERRORS REPLACE THE ERROR
                        0850 * CHARACTER ON THE SCREEN WITH A QUESTION MARK.
                        0851 *
C37E 21 00 00           0852 SHEX   LXI    H,0     CLEAR H & L
C381 1A                 0853 SHE1   LDAX   D       GET CHARACTER
C382 FE 20              0854        CPI    20H     IS IT A SPACE?
C384 C8                 0855        RZ     .       IF SO
C385 FE 2F              0856        CPI    '/'
C387 C8                 0857        RZ
C388 FE 3A              0858        CPI    ':'
C38A C8                 0859        RZ
                        0860 *
C38B 29                 0861 HCONV  DAD    H       MAKE ROOM FOR THE NEW ONE
C38C 29                 0862        DAD    H
C38D 29                 0863        DAD    H
C38E 29                 0864        DAD    H
C38F CD 9B C3           0865        CALL   HCOV1   DO THE CONVERSION
C392 D2 6B C4           0866        JNC    ERR1    NOT VALID HEXIDECIMAL VALUE
C395 85                 0867        ADD    L
C396 6F                 0868        MOV    L,A     MOVE IT IN
C397 13                 0869        INX    D       BUMP THE POINTER
C398 C3 81 C3           0870        JMP    SHE1
                        0871 *
C39B D6 30              0872 HCOV1  SUI    48      REMOVE ASCII BIAS
C39D FE 0A              0873        CPI    10
C39F D8                 0874        RC     .       IF LESS THAN 9
C3A0 D6 07              0875        SUI    7       IT'S A LATTER??
C3A2 FE 10              0876        CPI    10H
C3A4 C9                 0877        RET    .       WITH TEST IN HAND
                        0878 *
                        0879 *
                        0880 *  THIS ROUTINE WILL SEE IF A FIELD (OPERAND) IS PRESENT.
                        0881 *  IF NOT, THEN HL WILL REMAIN AS THEY WERE ON ENTRY.
                        0882 *  IF IT WAS PRESENT, THEN HL=THAT VALUE IN HEX.
                        0883 *
     C3A5               0884 PSCAN  EQU    $      OPTIONAL PIELD SCANNER
C3A5 CD 59 C3           0885        CALL   SBLK   SEE IF FIELD IS PRESENT
C3A8 C8                 0886        RZ     .      RETURN LEAVING HL AS THEY WERE ON ENTRY
C3A9 CD 7E C3           0887        CALL   SHEX   FIELD IS THERE, GO GET IT
C3AC C9                 0888        RET    .      HL= EITHER OPTIONAL FIELD (HEX), OR AS IT WAS
                        0889 *
                        0890 *
                        0891 *
                        0892 *
                        0893 *           DUMP COMMAND
                        0894 *
                        0895 *     THIS ROUTINE DUMPS CHARACTERS FROM MEMORY TO THE
                        0896 *  CURRENT OUTPUT DEVICE.
                        0897 *  ALL VALUES ARE DESPLAYED AS ASCII HEX.
                        0898 *
                        0899 *  THE COMMAND FORM IS AS FOLLOWS:
                        0900 *
                        0901 *        DUMP  ADDR1  ADDR2
                        0902 *
                        0903 *    THE VALUES FROM ADDR1 TO ADDR2 ARE THEN OUTPUT TO THE
                        0904 *  OUTPUT DEVICE.  IF ONLY ADDR1 IS SPECIFIED THEN THE
                        0905 *  VALUE AT THAT ADDRESS IS OUTPUT.
                        0906 *
                        0907 *  IF WHILE DUMPING, THE MODE KEY IS PRESSED, THE DUMP WILL
                        0908 *  BE TERMINATED.  IF THE SPACE BAR IS PRESSED, THE DUMP
                        0909 *  WILL BE TEMPORARILY SUSPENDED UNTIL ANY KEY IS PRESSED.
                        0910 *
     C3AD               0911 DUMP   EQU    $      SET UP REGS TO DUMP SPECIFIED AREA
C3AD CD 78 C3           0912        CALL   SCONV  GET START ADDR (REQUIRED)
C3B0 E5                 0913        PUSH   H      SAVE THE START ADDR
C3B1 CD A5 C3           0914        CALL   PSCAN  GET OPTIONAL END ADDR, HL=THIS OR START ADDR
C3B4 D1                 0915        POP    D      DE=START ADDR
C3B5 EB                 0916        XCHG   .      DE=END ADDR, HL=START ADDR NOW
                        0917 *
C3B6 CD 42 C3           0918 DLOOP  CALL   CRLF
C3B9 CD D9 C3           0919        CALL   ADOUT  OUTPUT ADDRESS
C3BC CD F7 C3           0920        CALL   BOUT   ANOTHER SPACE TO KEEP IT PRETTY
C3BF 0E 10              0921        MVI    C,16   VALUES PER LINE
                        0922 *
C3C1 7E                 0923 DLP1   MOV    A,M    GET THE CHR
C3C2 C5                 0924        PUSH   B      SAVE VALUE COUNT
C3C3 CD DE C3           0925        CALL   HBOUT  SEND IT OUT WITH A BLANK
C3C6 7C                 0926        MOV    A,H    CRNT ADDR
C3C7 BA                 0927        CMP    D      VERSUS ENDING ADDR
```

```
C3C8 DA D0 C3           0928        JC     DLP1A  NOT DONE YET
C3CB 7D                 0929        MOV    A,L    TRY LOW ORDER BYTE
C3CC BB                 0930        CMP    E
C3CD D2 18 C2           0931        JNC    COMND  ALL DONE WHEN CRNT REACHES ENDING
     C3D0               0932 DLP1A  EQU    $      HERE TO KEEP DUMPING
C3D0 C1                 0933        POP    B      VALUES PER LINE
C3D1 23                 0934        INX    H
C3D2 0D                 0935        DCR    C      BUMP THE LINE COUNT
C3D3 C2 C1 C3           0936        JNZ    DLP1   NOT ZERO IF MORE FOR THIS LINE
C3D6 C3 B6 C3           0937        JMP    DLOOP  DO A LFCR BEFORE THE NEXT
                        0938 *
                        0939 *    OUTPUT HL AS HEX 16 BIT VALUE
                        0940 *
C3D9 7C                 0941 ADOUT  MOV    A,H    H FIRST
C3DA CD FC C3           0942        CALL   HEOUT
C3DD 7D                 0943        MOV    A,L    THEN L FOLLOWED BY A SPACE
                        0944 *
C3DE CD FC C3           0945 HBOUT  CALL   HEOUT
C3E1 CD 1F C0           0946        CALL   SINP   SEE IF WE SHD ESCAPE FM DUMP
C3E4 CA F7 C3           0947        JZ     BOUT   NO--ADD THE SPACE THEN
C3E7 E6 7F              0948        ANI    7FH    MAKE SURE ITS CLEAR OF PARITY
C3E9 CA 18 C2           0949        JZ     COMND  EITHER MODE (OR CTL-@)
C3EC FE 20              0950        CPI    ' '    IS IT SPACE
C3EE C2 F7 C3           0951        JNZ    BOUT   NO--IGNORE THE CHAR
C3F1 CD 1F C0           0952 WTLP1  CALL   SINP   ON SPACE, WAIT FOR ANY OTHER CHAR
C3F4 CA F1 C3           0953        JZ     WTLP1  JUST LOOP AFTER A SPACE UNTIL ANY KEY PRESSED
C3F7 06 20              0954 BOUT   MVI    B,' '
C3F9 C3 19 C0           0955        JMP    SOUT   PUT IT OUT
                        0956 *
C3FC 4F                 0957 HEOUT  MOV    C,A    GET THE CHARACTER
C3FD 0F                 0958        RRC
C3FE 0F                 0959        RRC           MOVE THE HIGH FOUR DOWN
C3FF 0F                 0960        RRC
C400 0F                 0961        RRC
C401 CD 05 C4           0962        CALL   HEOU1  PUT THEM OUT
C404 79                 0963        MOV    A,C    THIS TIME THE LOW FOUR
                        0964 *
C405 E6 0F              0965 HEOU1  ANI    0FH    FOUR ON THE FLOOR
C407 C6 30              0966        ADI    48     WE WORK WITH ASCII HERE
C409 FE 3A              0967        CPI    58     0-9?
C40B DA 10 C4           0968        JC     OUTH   YUP!
C40E C6 07              0969        ADI    7      MAKE IT A LETTER
C410 47                 0970 OUTH   MOV    B,A    OUTPUT IT FROM REGISTER 'B'
C411 C3 19 C0           0971        JMP    SOUT
                        0972 *
                        0973 *
                        0974 *           ENTR COMMAND
                        0975 *
                        0976 *   THIS ROUTINE GETS VALUES FROM THE KEYBOARD AND ENTERS
                        0977 * THEM INTO MEMORY.  THE INPUT VALUES ARE SCANNED FOLLOWING
                        0978 * A STANDARD 'GCLIN' INPUT SO ON-SCREEN EDITING MAY TAKE
                        0979 * PLACE PRIOR TO THE LINE TERMINATOR.  A SLASH '/'
                        0980 * ENDS THE ROUTINE AND RETURNS CONTROL TO THE COMMAND MODE.
                        0981 *
C414 CD 78 C3           0982 ENTER  CALL   SCONV  SCAN OVER CHARS AND GET ADDRESS
C417 E5                 0983        PUSH   H      SAVE ADDRESS
                        0984 *
C418 CD 42 C3           0985 ENLOP  CALL   CRLF
C41B 06 3A              0986        MVI    B,':'
C41D CD 19 C0           0987        CALL   SOUT   DSPLY THE COLON
C420 CD 27 C2           0988        CALL   GCLI0  INIT AND PROCESS A LINE
C423 CD AA C2           0989        CALL   STUP   SET UP TO PROCESS INPUT LINE
C426 EB                 0990        XCHG   .      ....TO DE
                        0991 *
                        0992 *
C427 0E 03              0993 ENLO1  MVI    C,3    NO MORE THAN THREE SPACES BETWEEN VALUES
C429 CD 6E C3           0994        CALL   SCHR1  SCAN TO NEXT VALUE
C42C CA 18 C4           0995        JZ     ENLOP  LAST ENTRY FOUND START NEW LINE
                        0996 *
C42F FE 2F              0997        CPI    '/'    COMMAND TERMINATOR?
C431 CA 18 C2           0998        JZ     COMND  IF SO...
C434 CD 7E C3           0999        CALL   SHEX   CONVERT VALUE
C437 FE 3A              1000        CPI    ':'    ADDRESS TERMINATOR?
C439 CA 44 C4           1001        JZ     ENLO3  GO PROCESS IF SO
C43C 7D                 1002        MOV    A,L    GET LOW PART AS CONVERTED
C43D E1                 1003        POP    H      GET MEMORY ADDRESS
C43E 77                 1004        MOV    M,A    PUT IN THE VALUE
C43F 23                 1005        INX    H
C440 E5                 1006        PUSH   H      BACK GOES THE ADDRESS
C441 C3 27 C4           1007        JMP    ENLO1  CONTINUE THE SCAN
                        1008 *
C444 E3                 1009 ENLO3  XTHL   .      PUT NEW ADDRESS ON STACK
C445 13                 1010        INX    D      MOVE SCAN PAST TERMINATOR
C446 C3 27 C4           1011        JMP    ENLO1
                        1012 *
```

```
                    1013 *
                    1014 *              EXECUTE COMMAND
                    1015 *
                    1016 *   THIS ROUTINE GETS THE FOLLOWING PARAMETER AND DOES A
                    1017 * PROGRAM JUMP TO THE LOCATION GIVEN BY IT.  IF PROPER
                    1018 * STACK OPERATIONS ARE USED WITHIN THE EXTERNAL PROGRAM
                    1019 * IT CAN DO A STANDARD 'RET'URN TO THE CUTER COMMAND MODE.
                    1020 *
                    1021 *
C449 CD 78 C3       1022 EXEC   CALL   SCONV  SCAN PAST BLANKS AND GET PARAMETER
     C44C           1023 EXEC1  EQU    $      HERE TO GO TO HL
C44C E5             1024        PUSH   H      SAVE ON STACK
C44D 21 00 C0       1025        LXI    H,START  LET USER KNOW WHERE WE ARE
C450 C9             1026        RET    .      AND OFF TO USER
                    1027 *
                    1028 *
                    1029 *
                    1030 *
                    1031 *   THIS ROUTINE GETS A NAME OF UP TO 5 CHARACTERS
                    1032 *  FROM THE INPUT STRING.  IF THE TERMINATOR IS A
                    1033 *  SLASH (/) THEN THE CHARACTER FOLLOWING IS TAKEN
                    1034 *  AS THE CASSETTE UNIT SPECIFICATION.
                    1035 *
                    1036 *
     C451           1037 NAME0  EQU    $      ENTER HERE TO SET HL TO THEAD
C451 21 1C C8       1038        LXI    H,THEAD  PT WHERE TO PUT NAME
C454 CD 59 C3       1039 NAME   CALL   SBLK   SCAN OVER TO FIRST CHRS
C457 06 06          1040        MVI    B,6
                    1041 *
C459 1A             1042 NAME1  LDAX   D      GET CHARACTER
C45A FE 20          1043        CPI    ' '    NO UNIT DELIMITER
C45C CA 80 C4       1044        JZ     NFIL
C45F FE 2F          1045        CPI    '/'    UNIT DELIMTTER
C461 CA 80 C4       1046        JZ     NFIL
C464 77             1047        MOV    M,A
C465 13             1048        INX    D      BUMP THE SCAN POINTER
C466 23             1049        INX    H
C467 05             1050        DCR    B
C468 C2 59 C4       1051        JNZ    NAME1  NAME IS OK, FALL THRU TO 'ERR1' IF NOT
                    1052 *
                    1053 *     CUTER ERROR HANDLER
                    1054 *
C46B EB             1055 ERR1   XCHG   .      GET SCAN ADDRESS
C46C 36 3F          1056 ERR2   MVI    M,'?'  FLAG THE ERROR
C46E 3A 07 C8       1057        LDA    OPORT  SEE IF VIA VDM DRIVER
C471 B7             1058        ORA    A
C472 CA 18 C2       1059        JZ     COMND  YES--VDM SCREEN NOW HAS THE ?
C475 CD 42 C3       1060        CALL   CRLF
C478 06 3F          1061        MVI    B,'?'  SET UP THE ????
C47A CD 19 C0       1062        CALL   SOUT   INDICATE INPUT NOT VALID
C47D C3 18 C2       1063        JMP    COMND  NOW READY FOR NEXT INPUT
                    1064 *
                    1065 *
                    1066 *
                    1067 *  HERE WE HAVE SCANNED OFF THE NAME.  ZERO FILL IN FOR
                    1068 *  NAMES LESS THAN FIVE CHARACTERS.
                    1069 *
C480 36 00          1070 NFIL   MVI    M,0    PUT IN AT LEAST ONE ZERO
C482 23             1071        INX    H
C483 05             1072        DCR    B
C484 C2 80 C4       1073        JNZ    NFIL   LOOP UNTIL B IS ZERO
                    1074 *
C487 FE 2F          1075        CPI    '/'    IS THERE A UNIT SPECIFICATION?
C489 3E 01          1076        MVI    A,1    PRETEND NOT
C48B C2 94 C4       1077        JNZ    DEFLT
C48E 13             1078        INX    D      MOVE PAST THE TERMINATOR
C48F CD 6C C3       1079        CALL   SCHR   GO GET IT
C492 D6 30          1080        SUI    '0'    REMOVE ASCII BIAS
                    1081 *
     C494           1082 DEFLT  EQU    $      CNVRT TO INTERNAL BIT FOR TAPE CONTROL
C494 E6 01          1083        ANI    1      JUST BIT ZERO
C496 3E 80          1084        MVI    A,TAPE1  ASSUME TAPE ONE
C498 C2 9C C4       1085        JNZ    STUNT  IF NON ZERO, IT IS ONE
C49B 1F             1086        RAR    .      ELSE MAKE IT TAPE TWO
C49C 32 54 C8       1087 STUNT  STA    FNUMF  SET IT IN
C49F C9             1088        RET
                    1089 *
                    1090 *
                    1091 *
                    1092 *   THIS ROUTINE PROCESSES THE XEQ AND GET COMMANDS
                    1093 *
                    1094 *
C4A0 3E             1095 TXEQ   DB     3EH    THIS BEGINS "MVI" OF THE "XRA" FOLLOWING
C4A1 AF             1096 TLOAD  XRA    A      A=0 TLOAD, A=AF (#0) THEN XEQ
C4A2 F5             1097        PUSH   PSW    SAVE FLAG TO SAY WHETHER LOAD OR XEQ
```

```
C4A3 21 2C C8          1098        LXI    H,DHEAD  PLACE DUMMY HDR HERE FOR COMPARES
C4A6 CD 54 C4          1099        CALL   NAME   SET IN NAME AND UNIT
C4A9 21 00 00          1100        LXI    H,0    ASSUME LOAD ADDR NOT GIVEN
C4AC CD A5 C3          1101        CALL   PSCAN  HL EITHER =0, OR OVERRIDE LOAD ADDR
                       1102 *
C4AF EB                1103 TLOA2  XCHG   .      PUT ADDRESS IN DE
C4B0 21 2C C8          1104        LXI    H,DHEAD  PT TO NORMAL HDR
C4B3 7E                1105        MOV    A,M    GET 1ST CHAR OF NAME
C4B4 B7                1106        ORA    A      IS THERE A NAME?
C4B5 C2 BB C4          1107        JNZ    TLOA3  YES--LOOK FOR IT
C4B8 21 1C C8          1108        LXI    H,THEAD  PT TO SAME HDR TO LOAD NEXT FILE
C4BB E5                1109 TLOA3  PUSH   H      SAVE PTR TO WHICH HDR TO USE
C4BC CD 44 C5          1110        CALL   ALOAD  GET UNIT AND SPEED
C4BF E1                1111        POP    H      RESTORE PTR TO PROPER HDR TO USE
C4C0 CD C7 C6          1112        CALL   RTAPE  READ IN THE TAPE
C4C3 DA 10 C5          1113        JC     TAERR  TAPE ERROR?
                       1114 *
C4C6 CD 4C C5          1115        CALL   NAOUT  PUT OUT THE HEADER PARAMETERS
C4C9 F1                1116        POP    PSW    RESTORE FLAG SAYING WHETHER IT WAS LOAD OR XEQ
C4CA B7                1117        ORA    A
C4CB C8                1118        RZ     .      AUTO XEQ NOT WANTED
C4CC 3A 22 C8          1119        LDA    HTYPE  CHECK TYPE
C4CF B7                1120        ORA    A      SET FLAGS
C4D0 FA 10 C5          1121        JM     TAERR  TYPE IS NON XEQ
C4D3 3A 21 C8          1122        LDA    THEAD+5
C4D6 B7                1123        ORA    A
C4D7 C2 10 C5          1124        JNZ    TAERR  THE BYTE MUST BE ZERO FOR AUTO XEQ
C4DA 2A 27 C8          1125        LHLD   XEQAD  GET THE TAPE ADDRESS
C4DD C3 4C C4          1126        JMP    EXEC1  AND GO OFF TO IT
                       1127 *
                       1128 *
                       1129 *
                       1130 *   THIS ROUTINE IS USED TO SAVE PROGRAMS AND DATA ON
                       1131 *   THE CASSETTE UNIT.
                       1132 *
                       1133 *
     C4E0              1134 TSAVE  EQU    $      SAVE MEMORY IMAGE TO TAPE
C4E0 CD 51 C4          1135        CALL   NAME0  GET NAME AND UNIT
C4E3 CD 78 C3          1136        CALL   SCONV  GET START ADDRESS
C4E6 E5                1137        PUSH   H      SAVE START ADDR FOR SIZE COMPUTATION LATER
C4E7 CD 78 C3          1138        CALL   SCONV  GET END ADDR (REQUIRED)
C4EA E3                1139        XTHL   .      HL=START ADDR NOW, STACK=END ADDR
C4EB E5                1140        PUSH   H      STACK =START FOLLOWED BY END
C4EC CD A5 C3          1141        CALL   PSCAN  SEE IF RETRIEVE FROM ADDR
C4EF 22 25 C8          1142        SHLD   LOADR  EITHER ACTUAL START, OR OVERRIDE INTO HDR
C4F2 E1                1143        POP    H      HL=START ADDR
C4F3 D1                1144        POP    D      DE=END ADDR
C4F4 E5                1145        PUSH   H      PUT START BACK ONTO STACK
C4F5 7B                1146        MOV    A,E    SIZE=END-START+1
C4F6 95                1147        SUB    L
C4F7 6F                1148        MOV    L,A
C4F8 7A                1149        MOV    A,D
C4F9 DE 00             1150        SBI    0      THIS EQUALS A SBB H
C4FB 94                1151        SUB    H      THIS IS NEEDED
C4FC 67                1152        MOV    H,A
C4FD 23                1153        INX    H
C4FE 22 23 C8          1154        SHLD   BLOCK  STORE THE SIZE
C501 E5                1155        PUSH   H      SAVE AS THE BLOCK SIZE
                       1156 *
C502 CD 44 C5          1157        CALL   ALOAD  GET UNIT AND SPEED
C505 21 1C C8          1158        LXI    H,THEAD  PT TO HEADER TO WRITE
C508 CD AB C7          1159        CALL   WHEAD  TURN TAPE ON, THEN WRITE HEADER
C50B D1                1160        POP    D      GET BACK THE SIZE
C50C E1                1161        POP    H      AND GET BACK THE ACTUAL START ADDR
C50D C3 8C C7          1162        JMP    WTAP1  WHITE THE BLK (W/EXTRA PUSH)
                       1163 *
                       1164 *   OUTPUT ERROR AND HEADER
                       1165 *
C510 CD 42 C3          1166 TAERR  CALL   CRLF
C513 16 06             1167        MVI    D,6
C515 21 21 C5          1168        LXI    H,ERRM
C518 CD 66 C5          1169        CALL   NLOOP  OUTPUT ERROR
C51B CD 4C C5          1170        CALL   NAOUT  THEN THE HEADER
C51E C3 15 C2          1171        JMP    COMN1
                       1172 *
C521 45 52 52 4F       1173 ERRM   ASC    !ERROR !
     52 20
                       1174 *
                       1175 *
                       1176 *              CAT COMMAND
                       1177 *
                       1178 *   THIS ROUTINE READS HEADERS FROM THE TAPE AND OUTPUTS
                       1179 *   THEM TO THE OUTPUT DEVICE.  IT CONTINUES UNTIL THE
                       1180 *   MODE KEY IS DEPRESSED.
                       1181 *
```

```
     C527              1182 TLIST  EQU    $      PRODUCE A LIST OF FILES ON A TAPE
C527 CD 51 C4          1183        CALL   NAME0  GET UNIT IF ANY (NAME IS IGNORED)
C52A CD 42 C3          1184        CALL   CRLF   START ON A FRESH LINE
                       1185 *
                       1186 *
C52D CD 44 C5          1187 LLIST  CALL   ALOAD
C530 06 01             1188        MVI    B,1
C532 CD EB C7          1189        CALL   TON    TURN ON THE TAPE
C535 CD 1F C7          1190 LIST1  CALL   RHEAD
C538 DA 15 C2          1191        JC     COMN1  TRUN OFF THE TAPE UNIT
C53B C2 35 C5          1192        JNZ    LIST1
C53E CD 4C C5          1193        CALL   NAOUT  OUTPUT THE HEADER
C541 C3 2D C5          1194        JMP    LLIST
                       1195 *
                       1196 *
                       1197 *   THIS ROUTINE GETS THE CASSETTE UNIT NUMBER AND
                       1198 *   SPEED TO REGISTER "A" FOR THE TAPE CALLS
                       1199 *
C544 21 54 C8          1200 ALOAD  LXI    H,FNUMF  POINT TO THE UNIT SPECIFICATION
C547 3A 0D C8          1201        LDA    TSPD   GET THE TAPE SPEED
C54A B6                1202        ORA    M      PUT THEM TOGETHER
C54B C9                1203        RET    .      AND GO BACK
                       1204 *
                       1205 *   THIS ROUTINE OUTPUTS THE NAME AND PARAMETERS OF
                       1206 *   THEAD TO THE OUTPUT DEVICE.
                       1207 *
                       1208 *
C54C 16 08             1209 NAOUT  MVI    D,8
C54E 21 1B C8          1210        LXI    H,THEAD-1  POINT TO THE HEADER
C551 CD 66 C5          1211        CALL   NLOOP  OUTPUT THE HEADER
C554 CD F7 C3          1212        CALL   BOUT   ANOTHER BLANK
C557 2A 25 C8          1213        LHLD   LOADR  NOW THE LOAD ADDRESS
C55A CD D9 C3          1214        CALL   ADOUT  PUT IT OUT
C55D 2A 23 C8          1215        LHLD   BLOCK  AND THE BLOCK SIZE
C560 CD D9 C3          1216        CALL   ADOUT
C563 C3 42 C3          1217        JMP    CRLF   DO THE CRLF AND RETURN
                       1218 *
                       1219 *
C566 7E                1220 NLOOP  MOV    A,M    GET CHARACTER
C567 B7                1221        ORA    A
C568 C2 6D C5          1222        JNZ    CHRLI  IF IT ISN'T A ZERO
C56B 3E 20             1223        MVI    A,' '  SPACE OTHERWISE
     C56D              1224 CHRLI  EQU    $      CHAR IS OK TO SEND
C56D CD 10 C4          1225        CALL   OUTH   OUTPUT IT FROM A REG
C570 23                1226        INX    H
C571 15                1227        DCR    D
C572 C2 66 C5          1228        JNZ    NLOOP
C575 C9                1229        RET
                       1230 *
                       1231 *
                       1232 *
                       1233 *
                       1234 *      "SET" COMMAND
                       1235 *
                       1236 *   THIS ROUTINE GETS THE ASSOCIATED PARAMETER AND
                       1237 *   DISPATCHES TO THE PROPER ROUTINE FOR SETTING
                       1238 *   MEMORY VALUES.
                       1239 *
C576 CD 59 C3          1240 SET    CALL   SBLK   SCAN TO SECONDARY COMMAND
C579 CA 6B C4          1241        JZ     ERR1   MUST HAVE AT LEAST SOMETHING!!
C57C D5                1242        PUSH   D      SAVE SCAN ADDRESS
C57D CD 78 C3          1243        CALL   SCONV  CONVERT FOLLOWING VALUE
C580 E3                1244        XTHL   .      HL=SAVED SCAN ADDR AND STACK=VALUE
C581 11 11 C3          1245        LXI    D,SETAB  SECONDARY COMMAND TAALF
C584 CD 91 C2          1246        CALL   FDCOM  TRY TO LOCATE IT
C587 C3 82 C2          1247        JMP    DISP0  OFF TO IT OR ERROR IF NOT IN TBL
                       1248 *
                       1249 *
                       1250 *  THIS ROUTINE SETS THE TAPE SPEED
                       1251 *
     C58A              1252 TASPD  EQU    $      GET CONVERTED VALUE
C58A B7                1253        ORA    A      IS IT ZERO?
C58B CA 90 C5          1254        JZ     SETSP  YES--THAT IS A PROPER SPEED
C58E 3E 20             1255        MVI    A,32   NO--SET SPEED PROPERLY THEN
C590 32 0D C8          1256 SETSP  STA    TSPD
C593 C9                1257        RET
                       1258 *
                       1259 *
     C594              1260 STSPD  EQU    $      VDM ESCAPE SEQUENCE COMES HERE
C594 78                1261        MOV    A,B    GET CHAR FOR FOLLOWING DISPD
     C595              1262 DISPD  EQU    $      SET DISPLAY SPEED
C595 32 0B C8          1263        STA    SPEED
C598 C9                1264        RET
                       1265 *
                       1266 *
```

```
     C599              1267 SETIN  EQU    $      SET AN INPUT PSUEDO PORT
C599 32 06 C8          1268       STA    IPORT
C59C C9                1269       RET
                       1270 *
                       1271 *
     C59D              1272 SETOT  EQU    $      SET AN OUTPUT PSUEDO PORT
C59D 32 07 C8          1273       STA    OPORT
C5A0 C9                1274       RET
                       1275 *
                       1276 *
     C5A1              1277 SETCI  EQU    $      DEFINE USER INPUT RTN ADDR
C5A1 22 00 C8          1278       SHLD   UIPRT
C5A4 C9                1279       RET
                       1280 *
                       1281 *
     C5A5              1282 SETCO  EQU    $      DEFINE USER OUTPUT RTN ADDR
C5A5 22 02 C8          1283       SHLD   UOPRT
C5A8 C9                1284       RET
                       1285 *
                       1286 *
     C5A9              1287 SETTY  EQU    $      SET TAPE HDR TYPE
C5A9 32 22 C8          1288       STA    HTYPE
C5AC C9                1289       RET
                       1290 *
                       1291 *
     C5AD              1292 SETXQ  EQU    $      SET TAPE-EXECUTE ADDDR FOR HDR
C5AD 22 27 C8          1293       SHLD   XEQAD
C5B0 C9                1294       RET
                       1295 *
                       1296 *
     C5B1              1297 SETNU  EQU    $      HERE TO SET NUMBER OF NULLS
C5B1 32 10 C8          1298       STA    NUCNT  THIS IS IT
C5B4 C9                1299       RET    .
                       1300 *
                       1301 *
     C5B5              1302 SETCR  EQU    $      SET CRC TO BE NORMAL, OR IGNORE CRC ERRORS
C5B5 32 11 C8          1303       STA    IGNCR  FF=IGNORE CRC ERRORS, ELSE=NORMAL
C5B8 C9                1304       RET    .
                       1305 *
                       1306 *
     C5B9              1307 CUSET  EQU    $      TRY TO SET/CLEAR CUSTOM ROUTINE ADDR
C5B9 CD 51 C4          1308       CALL   NAME0  GET A NAME (S/B 2 CHARS OR MORE)
C5BC 21 18 C2          1309       LXI    H,COMND  PT HERE IN CASE ADDR NOT GIVEN
C5BF CD A5 C3          1310       CALL   PSCAN  GET OPTIONAL OPERAND IF ANY
C5C2 E5                1311       PUSH   H      SAVE THAT VALUE (IF ANY)
C5C3 21 1C C8          1312       LXI    H,THEAD  PT TO NAME
C5C6 CD 8E C2          1313       CALL   FDCOU  SEE IF NAME IS KNOWN IN CUST TABLE
C5C9 CA CF C5          1314       JZ     CUSE2  NO--PROCEED TO KNOW IT
C5CC 1B                1315       DCX    D    DE PT TO 1ST CHAR OF NAME IN TBL
C5CD 36 00             1316       MVI    M,0  (HL CAME BACK PT'ING TO THEAD)  CLR THIS NAME
     C5CF              1317 CUSE2  EQU    $    ENTER NEW ONE IN TBL
C5CF 7E                1318       MOV    A,M  GET 1ST CHAR OF NAME
C5D0 12                1319       STAX   D    PUT NAME INTO TABLE
C5D1 13                1320       INX    D
C5D2 23                1321       INX    H
C5D3 7E                1322       MOV    A,M    GET 2ND CHAR OF NAME
C5D4 12                1323       STAX   D      NAME IS NOW POSTED
C5D5 13                1324       INX    D      PT TO 1ST BYTE OF ADDR
C5D6 E1                1325       POP    H      RESTORE SAVED RTN ADDR
C5D7 EB                1326       XCHG   .      DE=RTN ADDR, HL=THIS CU ENTRY
C5D8 73                1327       MOV    M,E    LO BYTE
C5D9 23                1328       INX    H
C5DA 72                1329       MOV    M,D    AND HI BYTE
C5DB C9                1330       RET    .      ALL DONE
                       1331 *
                       1332 *
                       1333 * -*-
                       9999       COPY   CUTER3/1                              3 OF 3
                       1334 *
                       1335 *
                       1336 *
                       1337 *
                       1338 *   THE FOLLOWING ROUTINES PROVIDE "BYTE BY BYTE" ACCESS
                       1339 *  TO THE CASSETTE TAPES ON EITHER A READ OR WRITE BASIS.
                       1340 *
                       1341 *  THE TAPE IS READ ONE BLOCK AT A TIME AND INDIVIDUAL
                       1342 *  TRANSFERS OF DATA HANDLED BY MANAGING A BUFFER AREA.
                       1343 *
                       1344 *  THE BUFFER AREA IS CONTROLLED BY A FILE CONTROL BLOCK
                       1345 *  (FCB) WHOSE STRUCTURE IS:
                       1346 *
                       1347 *
                       1348 *     7 BYTES FOR EACH OR THE TWO FILES STRUCTURED AS
                       1349 *   FOLLOWS:
                       1350 *
```

```
                    1351 *         1 BYTE -  ACCESS CONTROL   00 IF CLOSED
                    1352 *                                    FF IF READING
                    1353 *                                    FE IF WRITING
                    1354 *         1 BYTE -  READ COUNTER
                    1355 *         1 BYTE -  BUFFER POSITION POINTER
                    1356 *         2 BYTE -  CONTROL HEADER ADDRESS
                    1357 *         2 BYTE -  BUFFER LOCATION ADDRESS
                    1358 *
                    1359 *
                    1360 *
                    1361 *        THIS ROUTINE "OPENS" THE CASSETTE UNIT FOR ACCESS
                    1362 *
                    1363 *   ON ENTRY:  A - HAS THE TAPE UNIT NUMBER (1 OR 2)
                    1364 *              HL - HAS USER SUPPLIED HEADER FOR TAPE FILE
                    1365 *
                    1366 *
                    1367 *   NORMAL RETURN:   ALL REGISTERS ARE ALTERED
                    1368 *                    BLOCK TS READY FOR ACCESS
                    1369 *
                    1370 *   ERROR RETURN:    CARRY BIT IS SET
                    1371 *
                    1372 *   ERRORS:  BLOCK ALREADY OPEN
                    1373 *
                    1374 *
C5DC E5             1375 BOPEN  PUSH   H      SAVE HEADER ADDRESS
C5DD CD 2F C6       1376        CALL   LFCB   GET ADDRESS OF FILE CONTROL
C5E0 C2 F6 C5       1377        JNZ    TERE2  FILE WAS ALREADY OPEN
C5E3 36 01          1378        MVI    M,1    NOW IT IS
C5E5 23             1379        INX    H      POINT TO READ COUNT
C5E6 77             1380        MOV    M,A    ZERO
C5E7 23             1381        INX    H      POINT TO BUFFER CURSOR
C5E8 77             1382        MOV    M,A    PUT IN THE ZERO COUNT
                    1383 *
                    1384 *  ALLOCATE THE BUFFER
                    1385 *
C5E9 11 63 C8       1386        LXI    D,FBUF1  POINT TO BUFFER AREA
C5EC 3A 54 C8       1387        LDA    FNUMF  GET WHICH ONE WE ARE GOING TO USE
C5EF 82             1388        ADD    D
C5F0 57             1389        MOV    D,A    256 BIT ADD
                    1390 *
C5F1 C1             1391 UBUF   POP    B      HEADER ADDRESS
C5F2 B7             1392        ORA    A      CLEAR CARRY AND RETURN AFTER STORING PARAMS
C5F3 C3 B2 C6       1393        JMP    PSTOR  STORE THE VALUES
                    1394 *
                    1395 *    GENERAL ERROR RETURN POINTS FOR STACK CONTROL
                    1396 *
C5F6 E1             1397 TERE2  POP    H
C5F7 D1             1398 TERE1  POP    D
C5F8 AF             1399 TERE0  XRA    A      CLEAR ALL FLAGS
C5F9 37             1400        STC    .      SET ERROR
C5FA C9             1401        RET
                    1402 *
                    1403 *
C5FB 3D             1404 EOFER  DCR    A      SET MINUS FLAGS
C5FC 37             1405        STC    .      AND CARRY
C5FD D1             1406        POP    D      CLEAR THE STACK
C5FE C9             1407        RET    .      THE FLAGS TELL ALL
                    1408 *
                    1409 *
                    1410 *
                    1411 *
                    1412 *   THIS ROUTINE CLOSES THE FILE BUFFER TO ALLOW ACCESS
                    1413 *   FOR A DIFFERENT CASSETTE OR PROGRAM.  IF THE FILE
                    1414 *   OPERATIONS WERE "WRITE" THEN THE LAST BLOCK IS WRITTED
                    1415 *   OUT AND AN "END OF FILE" WRITTEN TO THE TAPE.  IF
                    1416 *   THE OPERATIONS WERE "READS" THEN THE FILE IS JUST
                    1417 *   MADE READY FOR NEW USE.
                    1418 *
                    1419 *   ON ENTRY:  A - HAS WHICH UNIT (1 OR 2)
                    1420 *
                    1421 *   ERROR RETURNS:  FILE WASN'T OPEN
                    1422 *
                    1423 *
C5FF CD 2F C6       1424 PCLOS  CALL   LFCB   GET CONTROL BLOCK ADDRESS
C602 C8             1425        RZ     .      WASN'T OPEN, CARRY IS SET FROM LFCB
C603 B7             1426        ORA    A      CLEAR CARRY
C604 3C             1427        INR    A      SET CONDITION FLAGS
C605 36 00          1428        MVI    M,0    CLOSE THE CONTROL BYTE
C607 C8             1429        RZ     .      WE WERE READING...NOTHING MORE TO DO
                    1430 *
                    1431 *    THE FILE OPERATIONS WERE "WRITES"
                    1432 *
                    1433 *  PUT THE CURRENT BLOCK ON THE TAPE
                    1434 *  (EVEN IF ONLY ONE BYTE!!)
                    1435 *  THEN WRITE AN END OF FILE TO THE TAPE
```

```
                    1436 *
                    1437 *
C608 23             1438        INX    H
C609 23             1439        INX    H
C60A 7E             1440        MOV    A,M    GET CURSOR POSITION
C60B 7E             1441
C60C CD BB C6       1442        CALL   PLOAD  BC GET HEADER ADDRESS, DE BUFFER ADDRESS
C60F C5             1443        PUSH   B      HEADER TO STACK
C610 21 07 00       1444        LXI    H,BLKOF  OFFSET TO BLOCK SIZE
C613 09             1445        DAD    B
C614 B7             1446        ORA    A      TEST COUNT
C615 CA 27 C6       1447        JZ     EOFW   NO BYTES...JUST WRITE EOF
                    1448 *
                    1449 *    WRITE LAST BLOCK
                    1450 *
C618 E5             1451        PUSH   H      SAVE BLOCK SIZE POINTER FOR EOF
C619 77             1452        MOV    M,A    PUT IN COUNT
C61A 23             1453        INX    H
C61B 36 00          1454        MVI    M,0    ZERO THE HIGHER BYTE
C61D 23             1455        INX    H
C61E 73             1456        MOV    M,E    BUFFER ADDRESS
C61F 23             1457        INX    H
C620 72             1458        MOV    M,D
C621 60             1459        MOV    H,B
C622 69             1460        MOV    L,C    PUT HEADER ADDRESS IN HL
C623 CD 78 C7       1461        CALL   WFBLK  GO WRITE IT OUT
C626 E1             1462        POP    H      BLOCK SIZE POINTER
                    1463 *
                    1464 *   NOW WRITE END OF FILE TO CASSETTE
                    1465 *
C627 AF             1466 EOFW   XRA    A    PUT IN ZEROS FOR SIZE:  EOF MARK IS ZERO BYTES
C628 77             1467        MOV    M,A
C629 23             1468        INX    H
C62A 77             1469        MOV    M,A
C62B E1             1470        POP    H      HEADER ADDRESS
C62C C3 78 C7       1471        JMP    WFBLK  WRITE IT OUT AND RETURN
                    1472 *
                    1473 *
                    1474 *
                    1475 *
                    1476 *   THIS ROUTINE LOCATES THE FILE CONTROL BLOCK POINTED TO
                    1477 *   BY REGISTER "A".  ON RETURN HL POINT TO THE CONTROL BYT
                    1478 *   AND REGISTER "A" HAS THE CONTROL WORD WITH THE FLAGS
                    1479 *   SET FOR IMMEDIATE CONDITION DECISIONS.
                    1480 *
                    1481 *
C62F 21 55 C8       1482 LFCB   LXI    H,FCBAS  POINT TO THE BASE OF IT
C632 1F             1483        RAR    .      MOVE THE 1 & 2 TO 0 & 1 LIKE COMPUTERS LIKE
C633 E6 01          1484        ANI    1      SMALL NUMBERS ARE THE RULE
C635 32 54 C8       1485        STA    FNUMF  CURRENT ACCESS FILE NUMBER
C638 CA 3E C6       1486        JZ     LFCB1  UNIT ONE (VALUE OF ZERO)
C63B 21 5C C8       1487        LXI    H,FCBA2  UNIT TWO--PT TO ITS FCB
     C63E           1488 LFCB1  EQU    $      HL PT TO PROPER FCB
C63E 7E             1489        MOV    A,M    PICK UP FLAGS FM FCB
C63F B7             1490        ORA    A      SET FLAGS BASED ON CONTROL WORD
C640 37             1491        STC           SET CARRY IN CASE OF IMMEDIATE ERROR RETURN
C641 C9             1492        RET
                    1493 *
                    1494 *
                    1495 *
                    1496 *
                    1497 *    READ TAPE BYTE ROUTINE
                    1498 *
                    1499 *    ENTRY:       -  A -  HAS FILE NUMBER
                    1500 *    EXIT: NORMAL -  A -  HAS BYTE
                    1501 *          ERROR
                    1502 *            CARRY SET     - IF FILE NOT OPEN OR
                    1503 *                            PREVIOUS OPERATIONS WERE WRITE
                    1504 *            CARRY & MINUS - END OF FILE ENCOUNTERED
                    1505 *
                    1506 *
                    1507 *
                    1508 *
C642 CD 2F C6       1509 RTBYT  CALL   LFCB   LOCATE THE FILE CONTROL BLOCK
C645 C8             1510        RZ     .      FILE NOT OPEN
C646 3C             1511        INR    A      TEST IF FF
C647 FA F8 C5       1512        JM     TERE0  ERROR WAS WRITING
C64A 36 FF          1513        MVI    M,-1   SET IT AS READ  (IN CASE IT WAS JUST OPENED)
C64C 23             1514        INX    H
C64D 7E             1515        MOV    A,M    GET READ COUNT
C64E E5             1516        PUSH   H      SAVE COUNT ADDRESS
C64F 23             1517        INX    H
C650 CD BB C6       1518        CALL   PLOAD  GET THE OTHER PARAMETERS
C653 E1             1519        POP    H
C654 B7             1520        ORA    A
```

```
C655 C2 71 C6          1521        JNZ    GTBYT  IF NOT EMPTY GO GET BYTE
                       1522 *
                       1523 *  CURSOR POSITION WAS ZERO...READ A NEW BLOCK INTO
                       1524 *  THE BUFFER.
                       1525 *
C658 D5                1526 RDNBLK PUSH   D      BUFFER POINTER
C659 E5                1527        PUSH   H      TABLE ADDRESS
C65A 23                1528        INX    H
C65B CD A2 C6          1529        CALL   PHEAD  PREPARE THE HEADER FOR READ
C65E CD C4 C6          1530        CALL   RFBLK  READ IN THE BLOCK
C661 DA F6 C5          1531        JC     TERE2  ERROR POP OFF STACK BEFORE RETURN
C664 E1                1532        POP    H
C665 7B                1533        MOV    A,E    LOW BYTE OF COUNT (WILL BE ZERO IF 256)
C666 B2                1534        ORA    D      SEE IF BOTH ARE ZERO
C667 CA FB C5          1535        JZ     EOFER  BYTE COUNT WAS ZERO....END OF FILE
C66A 73                1536        MOV    M,E    NEW COUNT ( ZERO IS 256 AT THIS POINT)
C66B 23                1537        INX    H      BUFFER LOCATION POINTER
C66C 36 00             1538        MVI    M,0
C66E 2B                1539        DCX    H
C66F 7B                1540        MOV    A,E    COUNT TO A
C670 D1                1541        POP    D      GET BACK BUFFER ADDRESS
                       1542 *
                       1543 *
                       1544 *
                       1545 *   THIS ROUTINE GETS ONE BYTE FROM THE BUFFER
                       1546 *  AND RETURNS IT IN REGISTER "A".  IF THE END
                       1547 *  OF THE BUFFER IS REACHED IT MOVES THE POINTER
                       1548 *  TO THE BEGINNING OF THE BUFFER FOR THE NEXT
                       1549 *  LOAD.
                       1550 *
C671 3D                1551 GTBYT  DCR    A      BUMP THE COUNT
C672 77                1552        MOV    M,A    RESTORE IT
C673 23                1553        INX    H
C674 7E                1554        MOV    A,M    GET BUFFER POSITION
C675 34                1555        INR    M      BUMP IT
                       1556 *
C676 83                1557        ADD    E
C677 5F                1558        MOV    E,A    DE NOW POINT TO CORRECT BUFFER POSITION
C678 D2 7C C6          1559        JNC    RT1
C67B 14                1560        INR    D
C67C 1A                1561 RT1    LDAX   D      GET CHARACTER FROM BUFFER
C67D B7                1562        ORA    A      CLEAR CARRY
C67E C9                1563        RET    .      ALL DONE
                       1564 *
                       1565 *
                       1566 *
                       1567 *
                       1568 *      THIS ROUTINE IS USED TO WRITE A BYTE TO THE FILE
                       1569 *
                       1570 *      ON ENTRY:   A -  HAS FILE NUMBER
                       1571 *                  B -  HAS DATA BYTE
                       1572 *
                       1573 *
C67F CD 2F C6          1574 WTBYT  CALL   LFCB   GET CONTROL BLOCK
C682 C8                1575        RZ     .      FILE WASN'T OPEN
C683 3C                1576        INR    A
C684 C8                1577        RZ     .      FILE WAS READ
C685 36 FE             1578        MVI    M,0FEH  SET IT TO WRITE
C687 23                1579        INX    H
C688 23                1580        INX    H
C689 78                1581        MOV    A,B    GET CHARACTER
C68A F5                1582        PUSH   PSW
C68B E5                1583        PUSH   H      SAVE CONTROL ADDRESS+2
                       1584 *
                       1585 *   NOW DO THE WRITE
                       1586 *
C68C CD BB C6          1587        CALL   PLOAD  BC GETS HEADER ADDR, DE BUFFER ADDRESS
C68F E1                1588        POP    H
C690 7E                1589        MOV    A,M    COUNT BYTE
C691 83                1590        ADD    E
C692 5F                1591        MOV    E,A
C693 D2 97 C6          1592        JNC    WT1
C696 14                1593        INR    D
C697 F1                1594 WT1    POP    PSW    CHARACTER
C698 12                1595        STAX   D      PUT CHR IN BUFFER
C699 B7                1596        ORA    A      CLEAR FLAGS
C69A 34                1597        INR    M      INCREMENT THE COUNT
C69B C0                1598        RNZ    .      RETURN IF COUNT DIDN'T ROLL OVER
                       1599 *
                       1600 *   THE BUFFER IS FULL.  WRITE IT TO TAPE AND RESET
                       1601 *  CONTROL BLOCK.
                       1602 *
C69C CD A2 C6          1603        CALL   PHEAD  PREPARE THE HEADER
C69F C3 78 C7          1604        JMP    WFBLK  WRITE IT OUT AND RETURN
                       1605 *
```

```
                    1606 *
                    1607 *
                    1608 *
                    1609 *  THIS ROUTINE PUTS THE BLOCK SIZE (256) AND BUFFER
                    1610 *  ADDRESS IN THE FILE HEADER.
                    1611 *
C6A2 CD BB C6       1612 PHEAD  CALL   PLOAD  GET HEADER AND BUFFER ADDRESSES
C6A5 C5             1613        PUSH   B      HEADER ADDRESS
C6A6 21 06 00       1614        LXI    H,BLKOF-1  PSTOR DOES AN INCREMENT
C6A9 09             1615        DAD    B      HL POINT TO BLOCKSIZE ENTRY
C6AA 01 00 01       1616        LXI    B,256
C6AD CD B2 C6       1617        CALL   PSTOR
C6B0 E1             1618        POP    H      HL RETURN WITH HEADER ADDRESS
C6B1 C9             1619        RET
                    1620 *
                    1621 *
C6B2 23             1622 PSTOR  INX    H
C6B3 71             1623        MOV    M,C
C6B4 23             1624        INX    H
C6B5 70             1625        MOV    M,B
C6B6 23             1626        INX    H
C6B7 73             1627        MOV    M,E
C6B8 23             1628        INX    H
C6B9 72             1629        MOV    M,D
C6BA C9             1630        RET
                    1631 *
                    1632 *
C6BB 23             1633 PLOAD  INX    H
C6BC 4E             1634        MOV    C,M
C6BD 23             1635        INX    H
C6BE 46             1636        MOV    B,M
C6BF 23             1637        INX    H
C6C0 5E             1638        MOV    E,M
C6C1 23             1639        INX    H
C6C2 56             1640        MOV    D,M
C6C3 C9             1641        RET
                    1642 *
                    1643 *
                    1644 *
                    1645 *
                    1646 *
                    1647 *   THIS ROUTINE SETS THE CORRECT UNIT FOR SYSTEM READS
C6C4 CD DA C7       1648 RFBLK  CALL   GTUNT  SET UP A=UNTT WITH SPEED
                    1649 *
                    1650 *
                    1651 *
                    1652 *
                    1653 *              TAPE READ ROUTINES
                    1654 *
                    1655 *     ON-ENTRY:     A HAS UNIT AND SPEED
                    1656 *                   HL POINT TO HEADER BLOCK
                    1657 *                   DE HAVE OPTIONAL PUT ADDRESS
                    1658 *
                    1659 *     ON EXIT:      CARRY IS SET IF ERROR OCCURED
                    1660 *                   TAPE UNITS ARE OFF
                    1661 *
                    1662 *
C6C7 D5             1663 RTAPE  PUSH   D      SAVE OPTIONAL ADDRESS
C6C8 06 03          1664        MVI    B,3    SHORT DELAY
C6CA CD EB C7       1665        CALL   TON
C6CD DB FB          1666        IN     TDATA  CLEAR THE UART FLAGS
                    1667 *
C6CF E5             1668 PTAP1  PUSH   H      HEADER ADDRESS
C6D0 CD 1F C7       1669        CALL   RHEAD  GO READ HEADER
C6D3 E1             1670        POP    H
C6D4 DA 02 C7       1671        JC     TERR   IF AN ERROR OR ESC WAS RECEIVED
C6D7 C2 CF C6       1672        JNZ    PTAP1  IF VALID HEADER NOT FOUND
                    1673 *
                    1674 *  FOUND A VALID HEADER NOW DO COMPARE
                    1675 *
C6DA E5             1676        PUSH   H      GET BACK AND RESAVE ADDRESS
C6DB 11 1C C8       1677        LXI    D,THEAD
C6DE CD CE C7       1678        CALL   DHCMP  COMPARE DE-HL HEADERS
C6E1 E1             1679        POP    H
C6E2 C2 CF C6       1680        JNZ    PTAP1
                    1681 *
                    1682 *
C6E5 D1             1683        POP    D      OPTIONAL "PUT" ADDRESS
C6E6 7A             1684        MOV    A,D
C6E7 B3             1685        ORA    E      SEE IF DE IS ZERO
C6E8 2A 23 C8       1686        LHLD   BLOCK  GET BLOCK SIZE
C6EB EB             1687        XCHG   .      ...TO DE
                    1688 *  DE HAS HBLOCK....HL HAS USER OPTION
C6EC C2 F2 C6       1689        JNZ    RTAP   IF DE WAS ZERO GET TAPE LOAD ADDRESS
C6EF 2A 25 C8       1690        LHLD   LOADR  GET TAPE LOAD ADDRESS
```

```
                    1691 *
                    1692 *
                    1693 *     THIS ROUTINE READS "DE" BYTES FROM THE TAPE
                    1694 *     TO ADDRESS HL.  THE BYTES MUST BE FROM ONE
                    1695 *     CONTIGUOUS PHYSICAL BLOCK ON THE TAPE.
                    1696 *
                    1697 *          HL HAS "PUT" ADDRESS
                    1698 *          DE HAS SIZE OF TAPE BLOCK
                    1699 *
C6F2 D5             1700 RTAP   PUSH   D      SAVE SIZE FOR RETURN TO CALLING PROGRAM
                    1701 *
     C6F3           1702 RTAP2  EQU    $      HERE TO LOOP RDING RLKS
C6F3 CD 11 C7       1703        CALL   DCRCT  DROP COUNT, B=LEN THIS BLK
C6F6 CA 0C C7       1704        JZ     RTOFF  ZERO=ALL DONE
                    1705 *
C6F9 CD 40 C7       1706        CALL   RHED1  READ THAT MANY BYTES
C6FC DA 02 C7       1707        JC     TERR   IF ERROR OR ESC
C6FF CA F3 C6       1708        JZ     RTAP2  RD OK--READ SOME MORE
                    1709 *
                    1710 *  ERROR RETURN
                    1711 *
C702 AF             1712 TERR   XRA    A
C703 37             1713        STC    .      SET ERROR FLAGS
C704 C3 0D C7       1714        JMP    RTOF1
                    1715 *
                    1716 *
C707 06 01          1717 TOFF   MVI    B,1
C709 CD ED C7       1718        CALL   DELAY
C70C AF             1719 RTOFF  XRA    A
C70D D3 FA          1720 RTOF1  OUT    TAPPT
C70F D1             1721        POP    D      RETURN BYTE COUNT
C710 C9             1722        RET
                    1723 *
                    1724 *
     C711           1725 DCRCT  EQU    $      COMMON RTN TO COUNT DOWN BLK LENGTHS
C711 AF             1726        XRA    A      CLR FOR LATER TESTS
C712 47             1727        MOV    B,A    SET THIS BLK LEN=256
C713 B2             1728        ORA    D      IS AMNT LEFT < 256
C714 C2 1C C7       1729        JNZ    DCRC2  NO--REDUCE AMNT BY 256
C717 B3             1730        ORA    E      IS ENTIRE COUNT ZERO
C718 C8             1731        RZ     ALL DONE--ZERO THIS CONDITIOL
C719 43             1732        MOV    B,E    SET THIS BLK LEN TO AMNT REMAININ(',
C71A 5A             1733        MOV    E,D    MAKE ENTIRE COUNT ZERO NOW
C71B C9             1734        RET    .      ALL DONE (NON-ZERO FLAG)
     C71C           1735 DCRC2  EQU    $      REDUCE COUNT BY 256
C71C 15             1736        DCR    D      DROP BY 256
C71D B7             1737        ORA    A      FORCE NON-ZERO FLAG
C71E C9             1738        RET           NON-ZERO=NOT DONE YET (BLK LEN=256)
                    1739 *
                    1740 *
                    1741 *   READ THE HEADER
                    1742 *
C71F 06 0A          1743 RHEAD  MVI    B,10   FIND 10 NULLS
C721 CD 59 C7       1744 RHEA1  CALL   STAT
C724 D8             1745        RC     .      IF ESCAPE
C725 DB FB          1746        IN     TDATA  IGNORE ERROR CONDITIONS
C727 B7             1747        ORA    A      ZERO?
C728 C2 1F C7       1748        JNZ    RHEAD
C72B 05             1749        DCR    B
C72C C2 21 C7       1750        JNZ    RHEA1  LOOP UNTIL 10 IN A ROW
                    1751 *
                    1752 *    WAIT FOR THE START CHARACTER
                    1753 *
C72F CD 6B C7       1754 SOHL   CALL   TAPIN
C732 D8             1755        RC     .      ERROR OR ESCAPE
C733 FE 01          1756        CPI    1      ARE WE AT THE 01 YET (START CHAR)
C735 DA 2F C7       1757        JC     SOHL   NO, BUT STIL ZEROES
C738 C2 1F C7       1758        JNZ    RHEAD  NO, LOOK FOR ANOTHER 10 NULLS
                    1759 *
                    1760 *    WE HAVE 10 (OR MORE) NULLS FOLLOWED IMMEDIATELY
                    1761 *    BY AN 01.  NOW READ THE HEADER.
                    1762 *
C73B 21 1C C8       1763        LXI    H,THEAD  POINT TO BUFFER
C73E 06 10          1764        MVI    B,HLEN  LENGTH TO READ
                    1765 *
     C740           1766 RHED1  EQU    $      RD A BLOCK INTO HL FOR B BYTES
C740 0E 00          1767        MVI    C,0    INIT THE CRC
     C742           1768 RHED2  EQU    $      LOOP HERE
C742 CD 6B C7       1769        CALL   TAPIN  GET A BYTE
C745 D8             1770        RC
C746 77             1771        MOV    M,A    STORE IT
C747 23             1772        INX    H      INCREMENT ADDRESS
C748 CD A4 C7       1773        CALL   DOCRC  GO COMPUTE THE CRC
C74B 05             1774        DCR    B      WHOLE HEADER YET?
C74C C2 42 C7       1775        JNZ    RHED2  DO ALL THE BYTES
```

```
                    1776 *
                    1777 *   THIS ROUTINE GETS THE NEXT BYTE AND COMPARES IT
                    1778 * TO THE VALUE IN REGISTER C.  THE FLAGS ARE SET ON
                    1779 * RETURN.
                    1780 *
C74F CD 6B C7       1781        CALL   TAPIN  GET CRC BYTE
C752 A9             1782        XRA    C      CLR CARRY AND SET ZERO IF MATCH, ELSE NON-ZERO
C753 C8             1783        RZ     .      CRC IS FINE
C754 3A 11 C8       1784        LDA    IGNCR  BAD CRC, SHD WE STILL ACCEPT IT
C757 3C             1785        INR    A      SEE IF IT WAS FF, IF FF THEN ZERO SAYS IGN ERR
                    1786 *   NOW, CRC ERR DETECTION DEPENDS ON IGNCR.
C758 C9             1787        RET
                    1788 *
                    1789 *    THIS ROUTINE GETS THE NEXT AVAILABLE BYTE FROM THE
                    1790 *  TAPE.  WHILE WAITING FOR THE BYTE THE KEYBOARD IS TESTED
                    1791 *  FOR AN ESC COMMAND.  IF RECEIVED THE TAPE LOAD IS
                    1792 *  TERMINATED AND A RETURN TO THE COMMAND MODE IS MADE.
                    1793 *
C759 DB FA          1794 STAT   IN     TAPPT  TAPE STATUS PORT
C75B E6 40          1795        ANI    TDR
C75D C0             1796        RNZ
C75E CD 1F C0       1797        CALL   SINP   CHECK INPUT
C761 CA 59 C7       1798        JZ     STAT   NOTHING THERE YET
C764 E6 7F          1799        ANI    7FH    CLEAR PARITY 1ST
C766 C2 59 C7       1800        JNZ    STAT   EITHER MODE OR CTL-@
C769 37             1801        STC    .      SET ERROR FLAG
C76A C9             1802        RET    .      AND RETURN
                    1803 *
                    1804 *
                    1805 *
C76B CD 59 C7       1806 TAPIN  CALL   STAT   WAIT UNTIL A CHARACTER IS AVAILABLE
C76E D8             1807        RC
                    1808 *
C76F DB FA          1809 TREDY  IN     TAPPT  TAPE STATUS
C771 E6 18          1810        ANI    TFE+TOE  DATA ERROR?
C773 DB FB          1811        IN     TDATA  GET THE DATA
C775 C8             1812        RZ     .      IF NO ERRORS
C776 37             1813        STC    .      SET ERROR FLAG
C777 C9             1814        RET
                    1815 *
                    1816 *
                    1817 *  THIS ROUTINE GETS THE CORRECT UNIT FOR SYSTEM WRITES
C778 CD DA C7       1818 WFBLK  CALL   GTUNT  SET UP A WITH UNIT AND SPEED
                    1819 *
                    1820 *
                    1821 *
                    1822 *       WRITE TAPE BLOCK ROUTINE
                    1823 *
                    1824 *   ON ENTRY:   A   HAS UNIT AND SPEED
                    1825 *               HL  HAS POINTER TO HEADER
                    1826 *
                    1827 *
     C77B           1828 WTAPE  EQU    $      HERE TO WRITE TAPE
C77B E5             1829        PUSH   H      SAVE HEADER ADDRESS
C77C CD AB C7       1830        CALL   WHEAD  TURN ON, THEN WRITE HDR
C77F E1             1831        POP    H
C780 11 07 00       1832        LXI    D,BLKOF  OFFSET TO BLOCK SIZE IN HEADER
C783 19             1833        DAD    D      HL POINT TO BLOCK SIZE
C784 5E             1834        MOV    E,M
C785 23             1835        INX    H
C786 56             1836        MOV    D,M    DE HAVE SIZE
C787 23             1837        INX    H
C788 7E             1838        MOV    A,M
C789 23             1839        INX    H
C78A 66             1840        MOV    H,M
C78B 6F             1841        MOV    L,A    HL HAVE STARTING ADDRESS
                    1842 *
                    1843 *    THIS ROUTINE WRITES ONE PHYSICAL BLOCK ON THE
                    1844 *  TAPE "DE" BYTES LONG FROM ADDRESS "HL".
                    1845 *
                    1846 *
     C78C           1847 WTAP1  EQU    $      HERE FOR THE EXTRA PUSH
C78C E5             1848        PUSH   H      A DUMMY PUSH FOR LATER EXIT
     C78D           1849 WTAP2  EQU    $      LOOP HERE UNTIL ENTIRE AMOUNT READ
C78D CD 11 C7       1850        CALL   DCRCT  DROP COUNT IN DE AND SET UP B W/LEN THIS BLK
C790 CA 07 C7       1851        JZ     TOFF   RETURNS ZERO IF ALL DONE
C793 CD BF C7       1852        CALL   WTBL   WRITE BLOCK FOR BYTES IN B (256)
C796 C3 8D C7       1853        JMP    WTAP2  LOOP UNTIL ALL DONE
                    1854 *
                    1855 *
C799 F5             1856 WRTAP  PUSH   PSW
C79A DB FA          1857 WRWAT  IN     TAPPT  TAPE STATUS
C79C E6 80          1858        ANI    TTBE   IS TAPE READY FOR A CHAR YET
C79E CA 9A C7       1859        JZ     WRWAT  NO--WAIT
C7A1 F1             1860        POP    PSW    YES--RESTORE CHAR TO OUTPUT
```

```
C7A2 D3 FB              1861        OUT    TDATA  SEND CHAR TO TAPE
                        1862 *
     C7A4               1863 DOCRC  EQU    $      A COMMON CRC COMPUTATION ROUTINE
C7A4 91                 1864        SUB    C
C7A5 4F                 1865        MOV    C,A
C7A6 A9                 1866        XRA    C
C7A7 2F                 1867        CMA
C7A8 91                 1868        SUB    C
C7A9 4F                 1869        MOV    C,A
C7AA C9                 1870        RET    .      ONE  BYTE NOW WRITTEN
                        1871 *
                        1872 *
                        1873 *   THIS ROUTINE WRITES THE HEADER POINTED TO BY
                        1874 *   HL TO THE TAPE.
                        1875 *
     C7AB               1876 WHEAD  EQU    $      HERE TO 1ST TURN ON THE TAPE
C7AB CD E9 C7           1877        CALL   WTON   TURN IT ON, THEN WRITE HEADER
C7AE 16 32              1878        MVI    D,50   WRITE 50 ZEROS
C7B0 AF                 1879 NULOP  XRA    A
C7B1 CD 99 C7           1880        CALL   WRTAP
C7B4 15                 1881        DCR    D
C7B5 C2 B0 C7           1882        JNZ    NULOP
                        1883 *
C7B8 3E 01              1884        MVI    A,1
C7BA CD 99 C7           1885        CALL   WRTAP
C7BD 06 10              1886        MVI    B,HLEN  LENGTH TO WRITE OUT
                        1887 *
C7BF 0E 00              1888 WTBL   MVI    C,0    RESET CRC BYTE
C7C1 7E                 1889 WLOOP  MOV    A,M    GET CHARACTER
C7C2 CD 99 C7           1890        CALL   WRTAP  WRITE IT TO THE TAPE
C7C5 05                 1891        DCR    B
C7C6 23                 1892        INX    H
C7C7 C2 C1 C7           1893        JNZ    WLOOP
C7CA 79                 1814        MOV    A,C    GET CRC
C7CB C3 99 C7           1895        JMP    WRTAP  PUT IT ON THE TAPE AND RETURN
                        1896 *
                        1897 *
                        1898 *   THIS ROUTINE COMPARES THE HEADER IN THEAD TO
                        1899 *   THE USER SUPPLIED HEADER IN ADDRESS HL.
                        1900 *   ON RETURN IF ZERO IS SET THE TWO NAMES COMPARED
                        1901 *
C7CE 06 05              1902 DHCMP  MVI    B,5
C7D0 1A                 1903 DHLOP  LDAX   D
C7D1 BE                 1904        CMP    M
C7D2 C0                 1905        RNZ
C7D3 05                 1906        DCR    B
C7D4 C8                 1907        RZ     .      IF ALL FIVE COMPARED
C7D5 23                 1908        INX    H
C7D6 13                 1909        INX    D
C7D7 C3 D0 C7           1910        JMP    DHLOP
                        1911 *
     C7DA               1912 GTUNT  EQU    $      SET A=SPEED + UNIT
C7DA 3A 54 C8           1913        LDA    FNUMF  GET UNIT
C7DD B7                 1914        ORA    A      SEE WHICH UNIT
C7DE 3A 0D C8           1915        LDA    TSPD   BUT 1ST GET SPEED
C7E1 C2 E6 C7           1916        JNZ    GTUN2  MAKE IT UNIT TWO
C7E4 C6 40              1917        ADI    TAPE2  THIS ONCE=UNIT 2, TWICE=UNIT 1
C7E6 C6 40              1918 GTUN2  ADI    TAPE2  UNIT AND SPEED NOW SET IN A
C7E8 C9                 1919        RET    .      ALL DONE
                        1920 *
C7E9 06 04              1921 WTON   MVI    B,4    SET LOOP DELAY  (BIT LONGER ON A WRITE)
     C7EB               1922 TON    EQU    $      HERE TO TURN A TAPE ON THEN DELAY
C7EB D3 FA              1923        OUT    TAPPT  GET TAPE MOVING, THEN DELAY
                        1924 *
C7ED 11 00 00           1925 DELAY  LXI    D,0
C7F0 1B                 1926 DLOP1  DCX    D
C7F1 7A                 1927        MOV    A,D
C7F2 B3                 1928        ORA    E
C7F3 C2 F0 C7           1929        JNZ    DLOP1
C7F6 05                 1930        DCR    B
C7F7 C2 ED C7           1931        JNZ    DELAY
C7FA C9                 1932        RET
                        1933 *
                        1934 *
                        1935 ***** -- END OF PROGRAM--
                        1936 *
                        1937 *
                        1938 *
                        1939 *
                        1940 *    S Y S T E M    E Q U A T E S
                        1941 *
                        1942 *
                        1943 *          VDM PARAMETERS
                        1944 *
     CC00               1945 VDMEM  EQU    0CC00H  VDM SCREEN MEMORY
```

```
                    1946 *
                    1947 *
                    1948 *            KEYBOARD SPECIAL KEY ASSIGNMENTS
                    1949 *
                    1950 *  THESE DEFINITIONS ARE DESIGNED TO ALLOW
                    1951 *  COMPATABILITY WITH SOLOS(TM).  THESE ARE THE
                    1952 *  SAME KEYS WITH BIT 7 (X'80') STRIPPED OFF.
                    1953 *
      001A          1954 DOWN   EQU    1AH     CTL Z
      0017          1955 UP     EQU    17H     CIL W
      0001          1956 LEFT   EQU    01H     CTL A
      0013          1957 RIGHT  EQU    13H     CTL S
      000B          1958 CLEAR  EQU    0BH     CTL K
      000E          1959 HOME   EQU    0EH     CTL N
      0000          1960 MODE   EQU    D0H     CTL-@
      005F          1961 BACKS  EQU    5FH     BACKSPACE
      000A          1962 LF     EQU    10
      000D          1963 CR     EQU    13
      0020          1964 BLANK  EQU    ' '
      0020          1965 SPACE  EQU    BLANK
      0018          1966 CX     EQU    'X'-40H
      001B          1967 ESC    EQU    1BH
                    1968 *
                    1969 *          PORT ASSIGNMENTS
                    1970 *
      0000          1971 STAPT  EQU    0       STATUS PORT GENERAL
      0001          1972 SDATA  EQU    1       SERIAL DATA
      0002          1973 PDATA  EQU    2       PARALLEL DATA
      0003          1974 KDATA  EQU    3       KEYBOARD DATA
      00C8          1975 DSTAT  EQU    0C8H    VDM CONTROL PORT
      00FA          1976 TAPPT  EQU    0FAH    TAPE STATUS PORT
      00FB          1977 TDATA  EQU    0FBH    TAPE DATA PORT
      00FF          1978 SENSE  EQU    0FFH    SENSE SWITCHES
                    1979 *
                    1980 *
                    1981 *
                    1982 *          BIT ASSIGNMENT MASKS
                    1983 *
      0001          1984 SCD    EQU    1       SERIAL CARRIER DETECT
      0002          1985 SDSR   EQU    2       SERIAL DATA SET READY
      0004          1986 SPE    EQU    4       SERIAL PARITY ERROR
      0008          1987 SFE    EQU    8       SERIAL FRAMING ERROR
      0010          1988 SOE    EQU    16      SERIAL OVERRUN ERROR
      0020          1989 SCTS   EQU    32      SERIAL CLEAR TO SEND
      0040          1990 SDR    EQU    64      SERIAL DATA READY
      0080          1991 STBE   EQU    128     SERIAL TRANSMITTER BUFFER EMPTY
                    1992 *
      0001          1993 KDR    EQU    1       KEYBOARD DATA READY
      0002          1994 PDR    EQU    2       PARALLEL DATA READY
      0004          1995 PXDR   EQU    4       PARALLEL DEVICE READY
      0008          1996 TFE    EQU    8       TAPE FRAMING ERROR
      0010          1997 TOE    EQU    16      TAPE OVERFLOW ERROR
      0040          1998 TDR    EQU    64      TAPE DATA READY
      0080          1999 TTBE   EQU    128     TAPE TRANSMITTER BUFFER EMPTY
                    2000 *
      0001          2001 SOK    EQU    1       SCROLL OK FLAG
                    2002 *
      0080          2003 TAPE1  EQU    80H     1=TURN TAPE ONE ON
      0040          2004 TAPE2  EQU    40H     1=TURN TAPE TWO ON
                    2005 *
                    2006 *
                    2007 *
                    2008 *
                    2009 *       S Y S T E M   G L O B A L    A R E A
                    2010 *
C800                2011        ORG    START+0800H   RAM STARTS JUST AFTER ROM
                    2012 *
      C800          2013 SYSRAM EQU    $       START OF SYSTEM RAM
      CBFF          2014 SYSTP  EQU    SYSRAM+3FFH  STACK WORKS FM TOP DOWN
                    2015 *
                    2016 *
                    2017 *   PARAMETERS STORED IN RAM
                    2018 *
C800                2019 UIPRT  DS     2       USER DEFINED INPUT RTN IF NON ZERO
C802                2020 UOPRT  DS     2       USER DEFINED OUTPUT RTN IF NON ZERO
C804                2021 DFLTS  DS     2       DEFAULT PSUEDO I/O PORTS
C806                2022 IPORT  DS     1       CRNT INPUT PSUEDO PORT
C807                2023 OPORT  DS     1       CRNT OUTPUT PSUEDO PORT
C808                2024 NCHAR  DS     1       CURRENT CHARACTER POSITION
C809                2025 LINE   DS     1       CURRENT LINE POSITION
C80A                2026 BOT    DS     1       BEGINNING OF TEXT DISPLACEMENT
C80B                2027 SPEED  DS     1       SPEED CONTROL BYTE
C80C                2028 ESCFL  DS     1       ESCAPE FLAG CONTROL BYTE
C80D                2029 TSPD   DS     1       CURRENT TAPE SPEED
C80E                2030 INPTR  DS     2       PTR TO NEXT CHAR POSITION IN INLIN
```

```
C810                2031 NUCNT  DS     1      NUMBER OF NULLS AFTER CRLF
C811                2032 IGNCR  DS     1      IGN CRC ERR FLAG, FF=IGN CRC ERRS, ELSE=NORMAL
                    2033 *
C812                2034        DS     10     ROOM FOR FUTURE EXPANSION
                    2035 *
                    2036 * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                    2037 *    T H I S   I S   T H E   R E A D E R   L A Y O U T    *
                    2038 * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                    2039 *
C81C                2040 THEAD  DS     5      NAME
C821                2041        DS     1      THIS BYTE MUST BE ZERO
C822                2042 HTYPE  DS     1      TYPE
C823                2043 BLOCK  DS     2      BLOCK SIZE
C825                2044 LOADR  DS     2      LOAD ADDRESS
C827                2045 XEQAD  DS     2      AUTO EXECUTE ADDRESS
C829                2046 HSPR   DS     3      SPARES
                    2047 *
     0010           2048 HLEN   EQU    $-THEAD  LENGTH OF HEADER
     0007           2049 BLKOF  EQU    BLOCK-THEAD  OFFSET TO BLOCK SIZE
C82C                2050 DHEAD  DS     HLEN   A DUMMY HDR FOR COMPARES WHILE RD'ING
                    2051 *
                    2052 *
C83C                2053 CUTAB  DS     6*4    ROOM FOR UP TO 6 CUSTOM USER COMMANDS
                    2054 *
                    2055 *
C854                2056 FNUMF  DS     1      FOR CURRENT FILE OPERATIONS
C855                2057 FCBAS  DS     7      1ST FILE CONTROL BLOCK
C85C                2058 FCBA2  DS     7      2ND FILE CONTROL BLOCK
C863                2059 FBUF1  DS     2*256  SYSTEM FILE BUFFER BASE
CA63                2060        DS     1      "BELL" (X'07') FLAGS START OF INPUT BFR
CA64                2061 INLIN  DS     80     ROOM FOR THE INPUT LINE
     CAB4           2062 USARE  EQU    $      START OF USER AREA
                    2063 *
                    2064 *   REMEMBER THAT THE STACK WORKS ITS WAY DOWN-FROM
                    2065 *   THE END OF THIS 1K RAM AREA.
                    2066 *
                    2067 * -*-
```

ADOUT	C3D9	AINP	C022	ALOAD	C544	AOUT	C01C
ARET	C1C3	ARET1	C1C5	ARET2	C1CA	BACKS	005F
BLANK	0020	BLKOF	0007	BLOCK	C823	BOPEN	C5DC
BOT	C80A	BOUT	C3F7	CHAR	C0B7	CHRLI	C56D
CLEAR	000B	CLERA	C1DC	CLIN1	C122	CLIN2	C117
CLINE	C11C	COMN1	C215	COMND	C218	COMTA	C2BD
CONT	C264	COPRC	C26A	CR	000D	CREM	C15E
CRLF	C342	CUR	C0F4	CURET	C1CE	CURSC	C0F2
CUSE2	C5CF	CUSET	C5B9	CUTAB	C83C	CX	0018
DCRC2	C71C	DCRCT	C711	DEFLT	C494	DELAY	C7ED
DFLTS	C804	DHCMP	C7CE	DHEAD	C82C	DHLOP	C7D0
DISP0	C282	DISP1	C28B	DISPD	C595	DISPT	C287
DLOOP	C3B6	DLOP1	C7F0	DLP1	C3C1	DLP1A	C3D0
DOCRC	C7A4	DOWN	001A	DSTAT	00CB	DUMP	C3AD
ENLO1	C427	ENLO3	C444	ENLOP	C418	ENTER	C414
EOFER	C5FB	EOFW	C627	ERAS1	C0FE	ERAS3	C111
ERR1	C46B	ERR2	C46C	ERRIT	C064	ERRM	C521
ERRO1	C06F	ERROT	C06B	ESC	001B	ESCFL	C80C
ESCS	C187	ESCSP	C190	EXEC	C449	EXEC1	C44C
FBUF1	C863	FCBA2	C85C	FCBAS	C855	FCLOS	C00A
FDCOM	C291	FDCOU	C28E	FNUMF	C854	FOPEN	C007
GCLI0	C227	GCLI1	C232	GCLI2	C25F	GCLI3	C261
GCLIN	C239	GOBAC	C08E	GORK	C09F	GTBYT	C671
GTUN2	C7E6	GTUNT	C7DA	HBOUT	C3DE	HCONV	C38B
HCOV1	C39B	HEOU1	C405	HEOUT	C3FC	HLEN	0010
HOME	000E	HSPR	C829	HTYPE	C822	IGNCR	C811
INIT	C001	INLIN	CA64	INPTR	C80E	IOPRC	C026
IPORT	C806	ITAB	C309	KDATA	0003	KDR	0001
KREA1	C035	LEFT	0001	LF	000A	LFCB	C62F
LFCB1	C63E	LINE	C809	LIST1	C535	LLIST	C52D
LOADR	C825	MODE	0000	NAME	C454	NAME0	C451
NAME1	C459	NAOUT	C54C	NCHAR	C808	NCOM	C2A3
NEXT	C0A3	NFIL	C480	NLOOP	C566	NUCNT	C810
NULOP	C7B0	NULOT	C350	OCHAR	C0BB	OK	C0E4
OPORT	C807	OTAB	C301	OUTH	C410	OUTPR	C02E
PARIT	C050	PAROT	C059	PBACK	C166	PCLOS	C5FF
PCR	C16F	PCUR	C137	PDATA	0002	PDOWN	C0EE
PDR	0002	PERSE	C0F8	PFSC	C181	PHEAD	C6A2
PHOME	C108	PLEFT	C133	PLF	C175	PLOAD	C6BB
PRIT	C13D	PROMP	C33A	PSCAN	C3A5	PSTOR	C6B2
PTAP1	C6CF	PUP	C12C	PXDR	0004	RDSLK	C013
RDBYT	C00D	RDNBL	C658	RETRN	C004	RFBLK	C6C4
RHEA1	C721	RHEAD	C71F	RHED1	C740	RHED2	C742
RIGHT	0013	RT1	C67C	RTAP	C6F2	RTAP2	C6F3
RTAPE	C6C7	RTBYT	C642	RTOF1	C70D	RTOFF	C70C
SBLK	C359	SBLK1	C35B	SCD	0001	SCHR	C36C
SCHR1	C36E	SCONV	C378	SCROL	C0CF	SCTS	0020

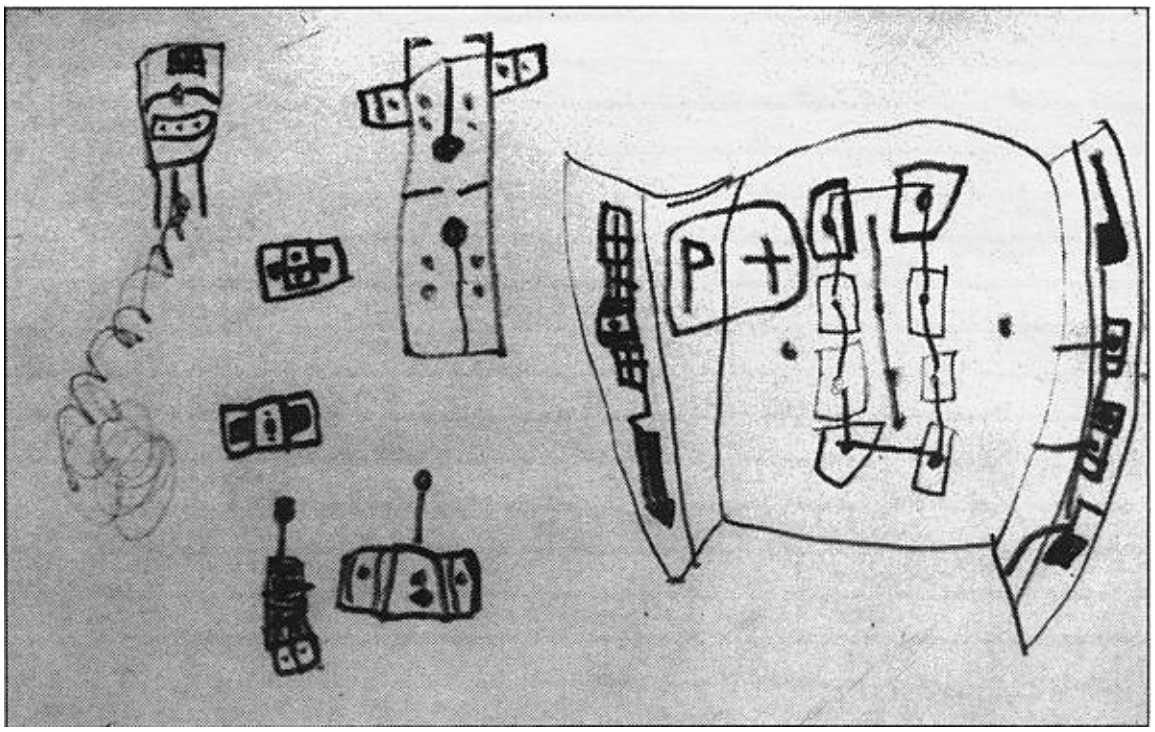
```
SDATA   0001    SDR     0040    SDSR    0002    SECON   C1B8
SENSE   00FF    SEROT   C046    SET     C576    SETAB   C311
SETCI   C5A1    SETCO   C5A5    SETCR   C5B5    SETIN   C599
SETNU   C5B1    SETOT   C59D    SETSP   C590    SETTY   C5A9
SETX    C1B0    SETXQ   C5AD    SETY    C1B4    SFE     0008
SHE1    C381    SHEX    C37E    SINP    C01F    SOE     0010
SOHL    C72F    SOK     0001    SOUT    C019    SPACE   0020
SPE     0004    SPEED   C80B    SREA1   C03E    SROL    C0D3
STAPT   0000    START   C000    STAT    C759    STBE    0080
STRTA   C1D7    STRTB   C1F4    STRTC   C1FF    STRTD   C20F
STSPD   C594    STUNT   C49C    STUP    C2AA    SYSRA   C800
SYSTP   CBFF    TAERR   C510    TAPE1   0080    TAPE2   0040
TAPIN   C76B    TAPPT   00FA    TASPD   C58A    TBL     C2E2
TDATA   00FB    TDR     0040    TERE0   C5F8    TERE1   C5F7
TERE2   C5F6    TERR    C702    TEE     0008    THEAD   C81C
TIMER   C09A    TLIST   C527    TLOA2   C4AF    TLOA3   C4BB
TLOAD   C4A1    TOE     0010    TOFF    C707    TON     C7EB
TREDY   C76F    TSAVE   C4E0    TSPD    C80D    TSRCH   C0A5
TTBE    0080    TXEQ    C4A0    UBUF    C5F1    UIPRT   C800
UOPRT   C802    UP      0017    USARE   CAB4    VDAD    C14B
VDAD2   C148    VDADD   C144    VDM01   C077    VDMEM   CC00
WFBLK   C778    WHEAD   C7AB    WLOOP   C7C1    WRBLK   C016
WRBYT   C010    WRTAP   C799    WRWAT   C79A    WT1     C697
WTAP1   C78C    WTAP2   C78D    WTAPE   C77B    WTBL    C7BF
WTBYT   C67F    WTLP1   C3F1    WTON    C7E9    XEQAD   C827
```

Waiting for BASIC-5

A lot of people have asked why Sol BASIC-5 took so long to be released. The main reason is Processor's policy concerning the release of new products: we don't ship 'til we have the finalized version, and that means product PLUS documentation. We feel that our reputation is based on selling products that live up to the advertising claims, and we intend to live up to our reputation. (Note: we've never been forced to recall a product.)

We could have provided you with a version of our original BASIC-5 a long time ago, with a modification of only 12 instructions. But that wasn't what we advertised Sol BASIC-5 to be. And in getting it to be what we wanted, we kept thinking it would be neat to add just a couple more nifty features and then just one more and one more and . . . So as with all fanatics, one thing led to another 'til somebody remembered that this is a business and there are customers out there who could only stand so much "neat stuff" (especially if they had to wait until the year 2000).

Anyway, by now you will have received said BASIC-5, so load it in and run it for a while. We think you'll agree that it was almost worth the wait.



Computer of the Future

Drawing by Brian Marsh, Age 6.

Contributions Welcome!

PROCESSOR TECHNOLOGY
ACCESS.

Processor Technology Corp.
6200 Hollis Street
Emeryville, CA 94608

Bulk Rate
U.S. Postage
PAID
Permit No. 3729
Oakland, Calif.